DETECCIÓN AUTOMÁTICA DE ACORDES EMPLEANDO TÉCNICAS DE CARACTERIZACIÓN DE AUDIO Y *MACHINE LEARNING*

Automatic Chord Detection Using Audio Characterization Techniques and Machine Learning

RAFAEL ALEJANDRO GIL URREGO

DIRECTOR: JUAN DAVID MARTINEZ VARGAS PhD. En Ingeniería

CODIRECTOR: LINA MARIA SEPULVEDA CANO PhD. En Ingeniería

UNIVERSIDAD EAFIT
ESCUELA DE CIENCIAS APLICADAS E INGENIERÍA
MAESTRIA EN CIENCIA DE DATOS Y ANALÍTICA
MEDELLÍN-ANTIOQUIA
2024

UNIVERSIDAD EAFIT

DETECCIÓN AUTOMÁTICA DE ACORDES EMPLEANDO TÉCNICAS DE CARACTERIZACIÓN DE AUDIO Y MACHINE LEARNING

Automatic Chord Detection Using Audio Characterization Techniques and Machine Learning

RAFAEL ALEJANDRO GIL URREGO

Trabajo de Grado presentado como requisito parcial para optar el título de Magister en Ciencia de Datos y Analítica.

UNIVERSIDAD EAFIT
MAESTRIA EN CIENCIA DE DATOS Y ANALÍTICA
MEDELLÍN-ANTIOQUIA
2024

TABLA DE CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

La detección automática de acordes en pistas de audio es esencial para el desarrollo de diversas aplicaciones musicales, como la transcripción musical y la generación de partituras. Por esta razón, ha surgido un creciente interés en el campo de la ciencia de datos para explorar diversas estrategias que resuelvan esta necesidad. El enfoque principal estudiado en los últimos años se basa en la extracción de características de los archivos de audio que contienen la información de los acordes. La transformación de la señal de audio a través de diferentes herramientas de análisis frecuencial ha generado datos con una mayor capacidad para describir las componentes musicales presentes en la pista de audio procesada. El espectrograma de Mel y el Cromograma son algunos de los métodos empleados para estas tareas. Además, se han utilizado modelos analíticos supervisados clásicos, como las Máquinas de Vectores de Soporte (*SVM*), *Random Forest* o Redes Neuronales Convolucionales (*CNN*) en varios estudios. Estos modelos han demostrado un alto nivel de precisión al momento de realizar la identificación de los acordes, aunque en la mayoría de los casos se han visto limitados en términos de la cantidad de las clases de acordes existentes a estimar, ya que un aumento en la cantidad de clases a estimar puede confundir el sistema, permitiendo un máximo de 24.

En este trabajo de grado, se desarrolló un sistema para la identificación automática de acordes musicales mediante la implementación de diferentes modelos analíticos clásicos y modernos. Para la extracción de características de audio, se emplearon los modelos pre-entrenados *HuBERT* y *VGGish*. Estas características fueron utilizadas como entrada para tres modelos clásicos, *SVM*, *Random Forest* y *Gradient Boosting*, con el fin de contrastar sus resultados con los obtenidos por un modelo moderno. La arquitectura HuBERT permite actuar como caracterizador así como clasificador por lo que fue la seleccionada como modelo moderno de contraste. Las pruebas se realizaron utilizando grabaciones de 48 clases diferentes de acordes, todas interpretadas en un piano digital, lo que proporcionó una base sólida para entrenar y evaluar el rendimiento del sistema propuesto.

El estudio confirmó las observaciones de investigaciones previas: para obtener estimaciones precisas de las clases de acordes, es fundamental mejorar las técnicas de caracterización de las grabaciones de audio de entrada. Un problema recurrente identificado fue la falta de descripción detallada de los componentes musicales en las grabaciones, lo que afectó la capacidad de los modelos para ofrecer resultados óptimos. Nuestros hallazgos resaltan que una extracción precisa de características es clave para reducir el error de generalización de los modelos, permitiendo una mejor identificación de las clases de acordes tanto en enfoques supervisados clásicos como en arquitecturas modernas como *HuBERT*.

Finalmente, se concluye que los modelos modernos, incluidos aquellos basados en *Transformers*, tienen una alta dependencia de la cantidad y diversidad de los datos. Para lograr una adaptabilidad efectiva, los datos de entrenamiento deben presentar suficientes variaciones dentro de una misma clase. Cuando los datos carecen de variabilidad intraclase, estos sistemas tienen dificultades para adaptarse a nuevas grabaciones que, aunque contienen la misma información musical, presentan ruido o distorsiones de fondo.

GLOSARIO

Acorde: se refiere a un determinado conjunto de notas diferentes que suenan al mismo tiempo o en sucesión rápida, constituyendo una identidad o unidad armónica. Las notas pueden pertenecer a la misma o a diferentes octavas (Recuero López , 1999).

Transcripción musical: la transcripción musical se define como el acto de escuchar una melodía y escribir la notación musical de las notas que la componen. En otras palabras, significa la transformación de una señal acústica en una representación simbólica que contiene las notas (con su altura y duración) y en la que se separan los distintos instrumentos empleados (Paez, 2014).

Procesamiento digital de señales: se refiere a la manipulación matemática de una señal que contiene cierto tipo de información para modificarla o mejorarla en algún sentido. El DSP tiene su fundamento en el dominio discreto del tiempo, el dominio de frecuencia discreta u otro dominio discreto permitiendo extraer la mayor cantidad de información útil de una señal para plasmarla en datos cuantificables (Alvarado Moya, 2011).

Aprendizaje de máquina (*machine learning*): se refiere a los métodos desarrollados por el área de la computación y la informática para dotar de inteligencia a los diferentes sistemas informáticos, estos métodos pretenden enseñar a partir de la experiencia a clasificar e identificar patrones que posteriormente una máquina pueda separar automáticamente (Smola & Vishwanathan, 2008).

Redes neuronales convolucionales: son modelos desarrollados en el campo de la ciencia de datos que imitan el funcionamiento de las redes neuronales humanas para procesar y analizar imágenes, sonidos, señales y otros tipos de datos complejos. Estos modelos aprenden a detectar patrones, como bordes, formas, ritmos y tonos, a través de capas especializadas, lo que permite a las máquinas clasificar y reconocer estos patrones de forma automática y con gran precisión (Zimmermann, 2006).

***Transformers*:** son un tipo de modelo desarrollado en el campo de la inteligencia artificial, diseñados para procesar secuencias de datos de forma eficiente. Estos modelos utilizan un mecanismo de atención para identificar relaciones entre elementos, permitiendo a las máquinas aprender y analizar información secuencialmente, como texto o imágenes con gran precisión (A. Vaswani, 2017).

OBJETIVOS

GENERAL

Construir un modelo de detección automático de acordes empleando técnicas de *machine learning* con el fin de evaluar e identificar las clases básicas de acordes empleadas dentro de la música popular occidental.

ESPECÍFICOS

- Explorar e identificar las características principales que definen a cada uno de los acordes básicos de la música occidental empleando el set de datos de audios de acordes interpretados en ambiente controlado.
- Emplear técnicas del *machine learning* para desarrollar clasificadores capaces de reconocer los acordes de una señal de audio procesada.
- Evaluar el desempeño de los modelos de aprendizaje de máquina con respecto a su capacidad de acertar en la clasificación de los acordes.

1. INTRODUCCIÓN

La detección automática de acordes es un problema que ha sido objeto de estudio en el campo de la ciencia de datos debido a su importancia en el desarrollo de aplicaciones relacionadas con la transcripción musical y la generación automática de partituras (Davy, 2007). Este problema se ha caracterizado por ser uno de los más desafiantes en área del procesamiento de señales de audio, en parte debido a la necesidad de procesar señales con una alta presencia de ruido de percusión, así como armónicos, subarmónicos y picos de frecuencia generados por la interacción polifónica de los instrumentos musicales, lo que distorsiona las características inherentes a la presencia de un acorde y añade información que dificulta su estimación (Benetos, 2013).

La capacidad de detectar acordes es una habilidad que se desarrolla a lo largo de una extensa formación musical y entrenamiento auditivo. No es exagerado afirmar que no todas las personas pueden adquirir esta habilidad y que no todos los músicos poseen la destreza necesaria para lograr una alta precisión en la identificación (Davy, 2007). Dada la complejidad del problema y la escasez de profesionales capacitados para llevar a cabo esta tarea de forma manual, es imperativo investigar nuevas técnicas que mejoren el rendimiento de las aplicaciones y satisfagan las necesidades de la comunidad musical.

Este trabajo de grado tiene como objetivo explorar nuevas técnicas analíticas para la extracción y reconocimiento de características con el propósito de establecer un marco de comparación entre las capacidades de los modelos clásicos y los modelos modernos basados en la arquitectura *Transformers* dado que actualmente es la arquitectura con mayores avances a nivel de aprendizaje automático. Además, buscamos abordar la clasificación automatizada de un amplio espectro de acordes al procesar un total de 48 clases, lo que constituye una tarea compleja. Esto se debe a las limitaciones previamente señaladas por Mauch en (Mauch, 2010), donde se advierte que los armónicos naturales de las notas principales de un acorde pueden poseer cantidades similares de energía en las bandas de frecuencia. Este fenómeno conlleva a una superposición de notas que distorsiona el vector de características. En otras palabras, un acorde no solo presentará características propias de su composición, sino que también podría incluir, en una proporción significativa, características de otros acordes cuya estructura comparta los mismos armónicos naturales de las notas principales ejecutadas.

La detección de acordes implica la extracción de características clave de una grabación de audio. En (S. H. Nawab, 2001) se propone el uso del "*Constant Q Spectral Transform"* (Transformación Espectral de Q Constante) (Brown, 1991) para calcular las bandas de frecuencia en las cuales la energía de las componentes frecuenciales principales de los acordes musicales es más prominente. Klapuri, por su parte, expande esta propuesta al introducir el concepto de "vector croma" o cromograma (Klapuri, 2009), el cual representa la energía en 12 bandas del espectro que componen una señal de audio. Estas 12 bandas de energía reflejan las 12 notas principales de la música popular occidental. Castro en (Castro-Ospina, 2024) propone el uso de una red neuronal pre entrenada la cual se encuentra especializada en el procesamiento de señales auditivas, esta misma, se encuentra inspirada en la arquitectura Grupo Visual Geométrico (*VGG* por sus siglas en inglés) que

originalmente fue desarrollada para realizar clasificación de imágenes, pero que con algunos ajustes, ha demostrado ser una herramienta efectiva para realizar procesamientos con archivos de audio y es denominada *VGGish*. El estudio de la caracterización se ha ido nutriendo de diferentes métodos como los mencionados anteriormente y han permitido mayor aplicación de los métodos de clasificación a señales complejas como lo es el estudio que se llevó a cabo en el desarrollo de este trabajo de grado.

En la construcción de un clasificador de acordes, se pueden emplear diversas técnicas de aprendizaje automático que ya han sido validadas en investigaciones previas. En este trabajo de grado, se adoptó el enfoque propuesto por Hsu (W.-N. Hsu, 2021), que combina mecanismos de atención, característicos de la arquitectura *Transformers* (A. Vaswani, 2017), con capas convolucionales de redes neuronales. Este método permite analizar señales de audio en múltiples escalas temporales y frecuenciales, capturando características relevantes en diferentes contextos y generando un vector de características robusto. La representación obtenida facilita una mejor estimación de las clases de acordes al crear clústeres que reflejan relaciones entre las características extraídas de la pieza musical procesada. Además, se integran modelos de aprendizaje supervisado ampliamente validados en la literatura, como las Máquinas de Vectores de Soporte, que son útiles para la separación de datos en espacios multidimensionales, logrando estimaciones de alta precisión mediante el uso del truco del *Kernel* (K. P. Bennett and J. Blue, 1998). También se emplean *Random Forests*, que son conjuntos de árboles de decisión capaces de capturar relaciones no lineales en los datos, lo que los hace especialmente adecuados para problemas con múltiples etiquetas (Breiman, 2001). Finalmente, se utilizan algoritmos de *Gradient Boosting*, como *XGBoost* o *LightGBM*, que mejoran la precisión de clasificación al combinar múltiples modelos simples en un ensamble, aportando gran utilidad para el reconocimiento preciso de datos etiquetados (S. Touzani, 1533–1543). Estos métodos clásicos sirven como contraste y validación de los resultados obtenidos con el modelo moderno.

En el desarrollo de nuestro clasificador automático de acordes, empleamos algoritmos clásicos así como dos pre-entrenados basados en redes neuronales. Empleamos dos métodos de extracción de características, uno a partir del modelo *VGGish* y el segundo empleando únicamente las capas de caracterización del modelo *HuBERT*, lo que nos permitió tener dos bases de aprendizaje para ser empleadas con los modelos clásicos, estos modelos de caracterización fueron aplicados a una base de datos que contiene grabaciones de 48 clases de acordes interpretados en un piano digital. El modelo *Hubert* no solo funciona como caracterizador, su arquitectura completa está desarrollada para lograr clasificaciones a partir de audios por lo que en si mismo funcionó tanto de caracterizador como de modelo de clasificación.

Para corroborar las capacidades discriminatorias de los modelos, se desarrolló una base de datos auxiliar que consta de 960 grabaciones de acordes con características diferentes como lo son instrumentos extra sonando dentro de la misma o sonidos de percusión que enriquecen el sonido pero aumentan la dificultad de detección. De esta manera, presentamos los resultados evaluados mediante métricas como *F1-score, Accuracy* y *Precision*, apropiadas para modelos de tipo supervisado (S. N. Alsubari, 2022), cada una de

estas métricas nos permite entender en términos de las predicciones, cuan acertado es el modelo dependiendo de sus éxitos o fallas, la métrica *F1-score* robustece los resultados al enfocarse no solo en los éxitos sino también en los falsos positivos o negativos. Nuestro objetivo es identificar las capacidades de estas herramientas y precisar las diferencias entre los modelos clásicos y los actuales.

2. MARCO TEÓRICO

2.1 Conceptos musicales

2.1.1 Notas musicales

En física una nota musical es relacionada como un sonido determinado por cierta vibración cuya frecuencia fundamental es constante. La música occidental se caracteriza por tomar sólo algunas de estas frecuencias y estandarizarlas, de esta forma, se obtienen 12 notas y 9 octavas que se puede observar en la Tabla 1, estas frecuencias representan los sonidos que el oído humano puede percibir y entender (Macías , 2012).

Tabla 1. Frecuencias de las notas musicales en Hz, octava -4 a 4

	-4	-3	-2	-1	0	1	2	3	4
Do		32,7	65,41	130,81	261,63	523,25	1046,5	2093	4186,01
Do#/Reb		34,65	69,3	138,59	277,18	554,37	1108,73	2217,46	4434,92
Re		36,71	73,42	146,83	293,66	587,33	1174,66	2349,32	4698,64
Re#/Mib		38,89	77,78	155,56	311,13	622,25	1244,51	2489,02	4978,03
Mi		41,2	82,41	164,81	329,63	659,26	1318,51	2637,02	5274,04
Fa	21,826	43,65	87,31	174,61	349,23	698,46	1396,91	2793,83	5587,65
Fa#/Solb	23,125	46,25	92,5	185	369,99	739,99	1479,98	2959,96	5919,91
Sol	24,5	49	98	196	392	783,99	1567,98	3135,96	6271,93
Sol#/Lab	25,96	51,91	103,83	207,65	415,3	830,61	1661,22	3322,44	
La	27,5	55	110	220	440	880	1760	3520	
La#/Sib	29,14	58,27	116,54	233,08	466	932,33	1864,66	3729,31	
Si	30,87	61,74	123,47	246,94	493,88	987,77	1975,53	3951,07	

La frecuencia fundamental de las notas es aproximadamente proporcional a un eje logarítmico frecuencial, es decir, si graficamos todas las frecuencias fundamentales de Do (Figura 1), tendríamos una gráfica logarítmica, de esta forma se puede establecer la relación de la ecuación 1.

$$f_{nota} = f_{ref} * 2^n \quad (1)$$

Donde:

- f_{nota} corresponde a la frecuencia de la nota en la octava deseada
- f_{ref} corresponde a la frecuencia de la nota en la octava 0
- n corresponde al número de la octava en la que se encuentra la nota

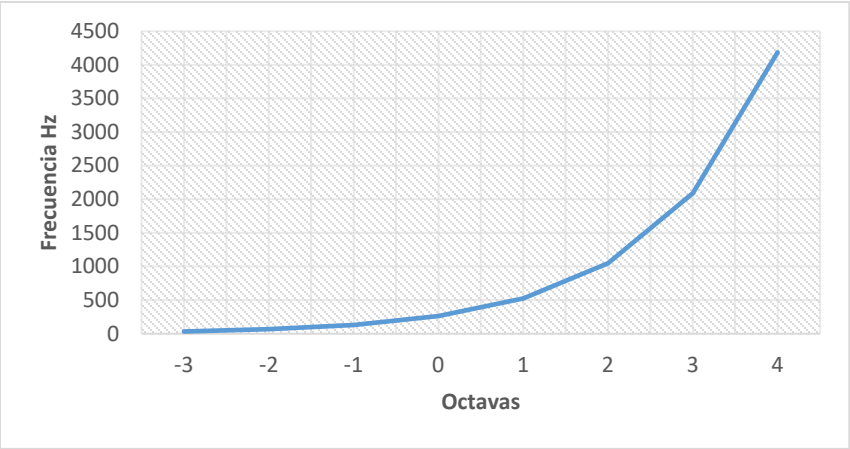

Figura 1. Relación frecuencia notas musicales vs octava en la que se encuentra, nota Do. Imagen propia

Cuando se emplean valores inferiores se habla de valores más graves, cuando se emplean valores superiores se entiende por valores más agudos, siendo *n=0* la frecuencia central.

Las 12 notas clasificadas se dividen en dos grupos:

- Notas diatónicas: Do, Re, Mi, Fa, Sol, La, Si
- Notas alteradas: son llamadas de esta forma porque representan una alteración de las notas diatónicas, y se caracterizan por agregar o quitar medio tono a la nota diatónica, son 5 notas alteradas en total las cuales son: Do#/Reb, Re#/Mib, Fa#/Solb, Sol#/Lab y La#/Sib (Macías , 2012)

2.1.2 Acordes

Un acorde se refiere a un determinado conjunto de notas diferentes que suenan al mismo tiempo o en sucesión rápida, constituyendo una identidad o unidad armónica, generalmente convergen armónicamente aunque actualmente, en la exploración de música vanguardista, los músicos han llegado a alterar el concepto de armonía, ingresando nuevos modos y estructuras a la acomodación de las notas creando “distorsiones armónicas” que enriquecen la estructura de la pieza interpretada. Las notas pueden pertenecer a una o diferentes octavas (Recuero López , 1999).

Se construyen comúnmente por intervalos de tercera. A las diferentes líneas de alturas de un fragmento polifónico se les llama voces, por tanto podemos hablar de las diferentes voces de un acorde (Levine, 2003); también se habla de factores del acorde. Una tríada común está compuesta por tres factores, una fundamental, su tercera y su quinta, que forma a su vez un intervalo de tercera con la tercera del acorde. (Segura Sogorb & Iñesta Quereda, 2015)

Los acordes se construyen a partir de su tónica y se clasifican en dos conjuntos dependiendo del modo en que se encuentren organizados las notas de la siguiente forma:

a. Modo mayor: a partir de la tónica, los dos tonos restantes se encuentran a una distancia de 4 y 7 semitonos respectivamente.

   Como ejemplo se presenta el acorde de Do Mayor que está conformado por las notas Do, Mi, Sol tal como se ve en la figura 2.

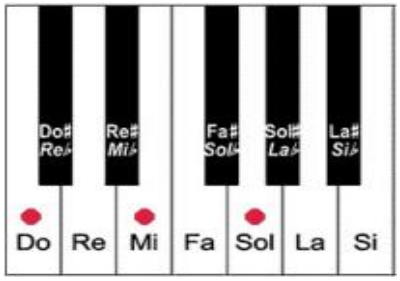

Figura 2. Acorde de Do Mayor en Piano. Extraído de: (Macías , 2012)

b. Modo menor: a partir de la tónica, los dos tonos restantes se encuentran a una distancia de 3 y 7 semitonos respectivamente.
   El acorde de Do menor está conformado por las notas Do, Re#/Mib, Sol tal como se ve en la figura 3.

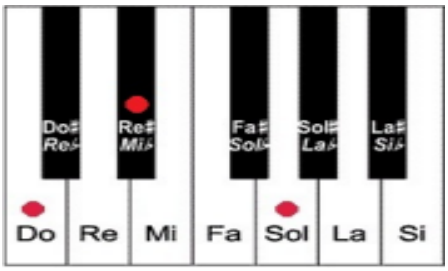

Figura 3. Acorde de Do Menor en Piano. Extraído de: (Macías , 2012)

2.2 Conceptos físicos y matemáticos

2.2.1 Transformada de Fourier

Para realizar el procesamiento de una señal de audio se debe tener en cuenta que debe ser capturada por un conversor analógico digital, al realizar este proceso, la señal no es más que una matriz de números que sin un adecuado tratamiento dejan de significar algo coherente. Fourier interpretó el proceso de discretización de señales y logró establecer una relación matemática a partir de las frecuencias que componen la función, a esta relación se la conoce como Transformada Discreta de Fourier (*DFT* por sus siglas en inglés) (Brown, 1991).

La *DFT* se encuentra regida por la frecuencia a la que se han tomado las muestras de la señal continua (la denominada frecuencia de muestreo, fs). La inversa de la frecuencia es el periodo de muestreo $Ts = 1/fs$, y se refiere al tiempo que transcurre entre cada dos

muestras consecutivas. En el mundo digital la magnitud referida al tiempo no es una entidad infinita por lo cual se encuentra representado como un índice $n \in \mathbb{N}$ de un vector en el que se van almacenando las muestras. Supongamos que la señal muestreada es un vector de N valores, $s\,[n]$, con $n \in [0, N-1]$. Cada muestra nos dirá el valor de la señal en nTs.

La ecuación 2 representa el cálculo matemático de la *DFT*.

$$S[k] = \sum_{n=0}^{N-1} s[n] e^{-j\frac{2\pi kn}{N}}, \quad k = 0, \dots, N-1 \tag{2}$$

Donde:

- $S[k]$ es el k-ésimo fasor de la combinación lineal
- N es el número de muestras tomados en un periodo de la señal
- $s[n]$ representa la n-ésima muestra de la señal en el dominio del tiempo

$S\,[k]$ es un número complejo y con él podemos conocer tanto el espectro de amplitudes como el de fases. Si se desea calcular el espectro de amplitudes, basta con utilizar una de las componentes (seno o coseno) presentado en la ecuación 3 (Segura Sogorb & Iñesta Quereda, 2015).

$$|S[k]| = \sum_{n=0}^{N-1} s[n] \sin\left(-\frac{2\pi kn}{N}\right), \quad k = 0, \dots, N-1 \tag{3}$$

Ecuación 1. Cálculo del componente de amplitud del espectro

Para ilustrar la *DFT*, en la figura 4 se presenta la gráfica de una señal senoidal con frecuencia 5 Hz y su componente en el dominio de la frecuencia.

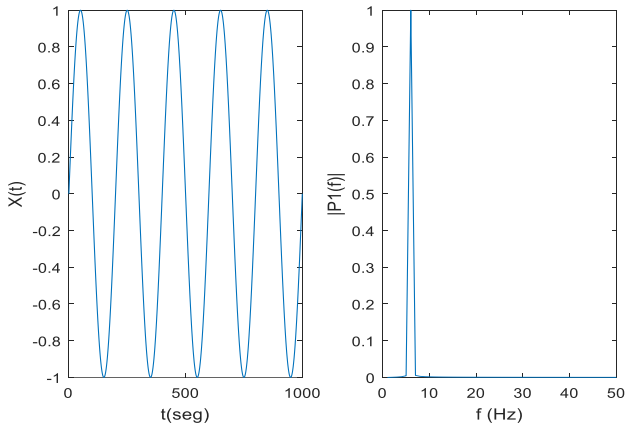

Figura 4. Señal senoidal de frecuencia 5Hz, gráfica dominio temporal y frecuencial. Imagen propia

2.2.2 Transformada Q

Aunque la Transformada de Fourier (*FT*) es una de las herramientas más empleadas en el procesamiento de señales, presenta una significativa limitante debido a que considera el análisis de una señal durante un intervalo de tiempo interminable, lo que resulta poco práctico debido a que la mayoría de las señales analógicas son finitas y presentan notables variaciones que dificultan la acción de periodicidad necesaria para extraer la información completa de la onda. Teóricamente la *FT* calcula el espectro de una señal infinita y estacionaria, pero como se dijo anteriormente, en la práctica los sonidos son finitos y cambiantes.

Bajo las condiciones cambiantes de las señales, los teóricos encontraron la forma de estudiarlas precisando fragmentos de la señal en los cuales la misma se presenta como invariable, en otras palabras, la señal se compone de pequeños fragmentos de esta que presentan periodo y frecuencia, siendo este el caso, la forma adecuada de hacer el procesamiento de una señal es empleando una función "ventana" $W[n-m]$ la cual no tendrá ningún efecto excepto en los instantes τ que se quieran estudiar. La multiplicación de esta función por la señal original (dicho proceso es conocido como enventanamiento de la señal) tendrá como resultado el fragmento de señal a transformar, mediante una nueva versión de la *FT* llamada transformada de Fourier a corto plazo (*short-time FT*, *STFT*) la cual se presenta en la ecuación 4 (Brown, 1991).

$$X[k,m] = \sum_{n=0}^{N-1} W[n-m]\, x[n] e^{-j2\pi kn/N} \tag{4}$$

Donde:

- $X[k,m]$ es el valor de la frecuencia en el intervalo n-m.
- $W[n-m]$ es el valor de la ventana.
- $x[n]$ es la señal para transformar.
- k es el índice de frecuencia, a medida que k aumenta, se obtiene información sobre componentes de frecuencia más altas de la señal.

Al generar los valores de frecuencia de la señal en instantes finitos, es posible obtener su espectro de frecuencias a partir del cálculo del espectrograma que es el resultado de elevar al cuadrado la magnitud de la *STFT*, tal como se presenta en la ecuación 5.

$$x(t) = |X[k,m]|^2 \tag{5}$$

Una de las limitaciones de la *STFT* es que emplea ventanas de longitud fija. Fijar dicho parámetro implica alcanzar cierto compromiso entre la resolución frecuencial y la temporal. Si se emplean ventanas pequeñas, las frecuencias con longitud de onda grande no pueden ser apreciadas, diferente a si usamos una ventana grande en este caso se obtiene una resolución temporal pobre, dicho efecto es denominado principio de incertidumbre. Una

alternativa a la *STFT* que resuelve medianamente este problema, que hace uso de ventanas de longitud dependiente a la frecuencia, es la transformada de Q constante o transformada Q. Esta transformada está estrechamente relacionada con la *FT* de la siguiente forma (Brown, 1991):

Para análisis musical, se pueden tomar los componentes de frecuencia correspondientes al cuarto de tono espaciado en la misma escala de temperamento. La frecuencia k del componente espectral se define como: $f_k = (2^{\frac{1}{24}})^k f_{min}$, donde f_{min} es la frecuencia más baja sobre la que se desea información.

Dada una serie de datos muestreados en *S=1/T*, siendo *T* el periodo de muestreo de los datos, para cada intervalo de frecuencia podemos definir lo siguiente:

- Frecuencia central del sistema: f
- Ancho de banda, indica el rango de frecuencias alrededor de f en el que el sistema sigue resonando de manera significativa: δf
- Factor de calidad: $Q = \frac{f}{\delta f}$
- Longitud de la ventana para el k-ésimo segmento: $N[k] = \frac{S}{\delta f_k} = \frac{S}{f_k} Q$

De esta forma, la ventana de procesamiento se encuentra en función de la frecuencia de la porción de señal.

- La potencia relativa de cada segmento tiende a decaer cuando se tratan frecuencias altas debido a que estas tienen menor cantidad de componentes, para compensarlo se hace la normalización del vector $N[k]$

- Cualquiera de las funciones ventana pueden ejercer dicha función, tomando como ejemplo la ventana *Hanning* tendremos:

$$W[k,n] = \alpha + (1-\alpha)\cos\frac{2\pi n}{N[k]}, \quad \alpha = 25\,/\,46, \;\; 0 \le n \le N[k]-1$$

- La frecuencia digital $\frac{2\pi k}{N}$, se convierte en: $\frac{2\pi Q}{N[k]}$

Obteniendo finalmente, la función caracterizada como transformada Q:

$$X[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} W[k,n]x[n]e^{\frac{-j2\pi Qn}{N[k]}} \tag{6}$$

Otra característica interesante de la transformada Q es que su resolución temporal es incrementada con la frecuencia. En la figura 5 se ejemplifica la forma en que operan la ST*FT* y la transformada Q con respecto a la resolución en frecuencia de la toma de muestras. Una situación similar se da en el sistema auditivo humano; al igual que el oído,

un sistema digital necesita más tiempo para percibir la frecuencia de un tono grave que para percibir la de un tono agudo. (Macías , 2012)

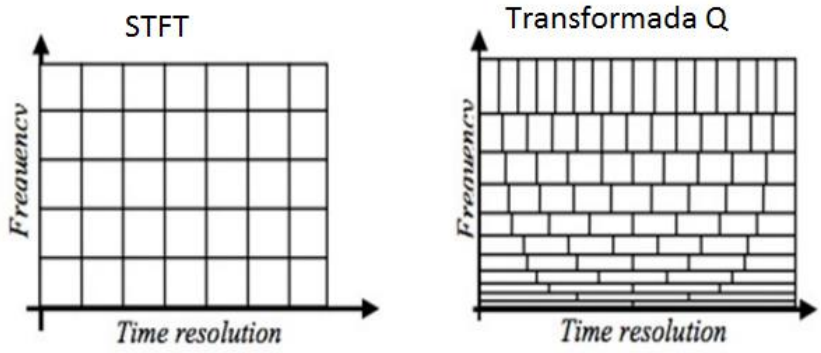

Figura 5.Comparación resolución espectral entre la STFT y la Transformada Q. Extraído de: (Macías , 2012)

2.2.3 Aprendizaje de máquina

Con la cantidad de datos que se manejan actualmente, el apogeo de las telecomunicaciones y la evolución de los sistemas, diferentes áreas del conocimiento han gestionado sistemas de aprendizaje que logran, a partir de ciertas reglas, clasificar y tomar decisiones autónomas con respecto a un grupo de información que le sea introducido, de esta forma, se ha dotado de cierta inteligencia a los sistemas permitiéndoles aprender ciertos patrones como lo haría un niño pequeño. La pieza que dota de inteligencia a la máquina es un algoritmo que revisa los datos y es capaz de prever comportamientos futuros automáticamente, de esta forma, los sistemas tienden a mejorar de forma autónoma con el tiempo (Jiang, 2022).

Existen diferentes tipos de algoritmos que han evolucionado con el tiempo y que se emplean en diferentes tipos de aplicaciones, en este trabajo de grado empleamos cuatro: máquinas de vectores de soporte, random forest, gradient boosting y un método actual basado en la arquitectura transformes denominado *HuBERT*, los conocimientos necesarios para entender el funcionamiento de cada uno de estos se encuentran a continuación.

2.2.3.1 Redes neuronales artificiales

Las redes neuronales artificiales nacen de emular el comportamiento biológico de las neuronas humanas, se pretende como tal, imitar la capacidad de memorizar y de asociar hechos por medio de la experiencia, es decir, se desarrolló un algoritmo capaz de aprender a partir del entrenamiento de características (Sánchez Anzola, 2015). Lo que básicamente ocurre en una neurona biológica es lo siguiente: la neurona es estimulada o excitada a través de sus entradas (*inputs*) y cuando se alcanza un cierto umbral, la neurona se dispara o activa, pasando una señal hacia el axón que comunica el estímulo a las neuronas contiguas a ella. Posteriores investigaciones condujeron al descubrimiento de que estos procesos son el resultado de eventos electroquímicos.

Para emular dicho proceso biológico se crea una red como se ve en la figura 6 que se compone básicamente por una capa de entrada la cual debe tener la misma cantidad de neuronas que la cantidad de señales que se le vayan a introducir, las capas ocultas que pueden tener la configuración que el diseñador crea conveniente, y la capa de salida que debe tener igual cantidad de neuronas que señales de salida. El tipo de red neuronal con el que se desarrolló el transcriptor es de carácter *Feed-Forward*, lo que hace referencia a un aprendizaje de un número estable de variables las cuales sirven como consultores a la hora de tomar la decisión necesaria, en contraste con un sistema realimentado, la red neuronal de tipo *Feed-Forward* no tiene capacidad para reajustar sus parámetros según el tipo de entrada que este percibiendo, por lo que las salidas siempre son fijas en cuanto al tipo de programación que haya realizado el diseñador (Sánchez Anzola, 2015).

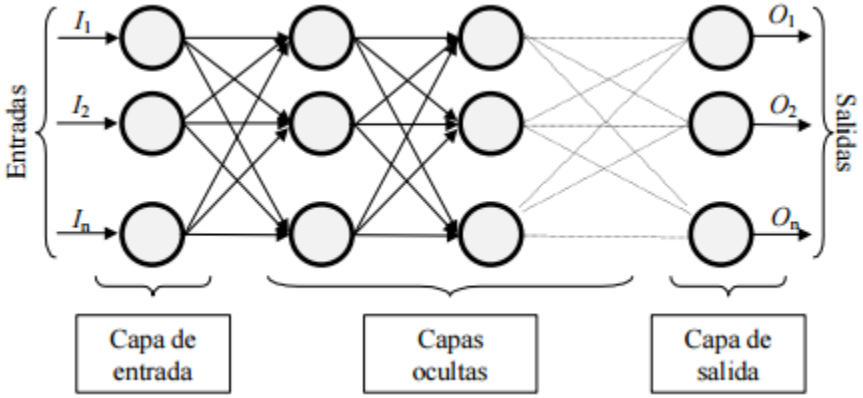

Figura 6. Componentes de una red neuronal artificial. Extraído de: (Macías , 2012)

La red neuronal artificial está conformada por tres elementos básicos los cuales se pueden apreciar en la figura 6, estos elementos son: la capa de entrada, que tiene como función ingresar los datos al sistema a partir de la cantidad de características que deba reconocer según el criterio del diseñador, las capas ocultas que se encargan de procesar cada una de las características y generar una "conciencia" del tipo de dato que se está evaluando para generar un posterior resultado, finalmente la capa de salida se encarga de alistar los datos procesados por la capa oculta y mostrarlos en un orden coherente.

2.2.3.2 Máquinas de Vectores de Soporte

Las Máquinas de Vectores de Soporte (*SVM* por su nombre en inglés *Support Vector Machines)* es una técnica de clasificación desarrollada por Vapnik y Cortés hacia el año 1995 (Cortes, 1995), debido a su robustez a la hora de enfrentar problemas de clasificación y regresión, además del auge de los sistemas dotados con "inteligencia artificial", han tenido un fuerte impacto al punto que, múltiples teóricos han descubierto en ellas una versatilidad lo suficientemente confiable para hacer desarrollos en torno a ellas. La teoría de la *SVM* está basada en la idea de minimización de riesgo estructural (*SRM*). En muchas aplicaciones, las *SVM* han mostrado tener gran desempeño, aventajando a otras máquinas de aprendizaje como los árboles de decisión y se han posicionado como herramientas de gran eficiencia para resolver problemas de clasificación (Betancourt , 2005).

La idea básica del funcionamiento de una *SVM* es transformar el plano de dimensionamiento de un conjunto de datos en uno superior aplicando una función *Kernel*, en donde dichos datos sean linealmente separables, lo cual resuelve el problema de clasificación sin importar la dimensión del plano en el que se encuentren. En general, la idea principal con la opera una *SVM* es, a través de los datos de entrada que le sean presentados en un principio, así como las etiquetas de las clases a la que pertenece cada uno de los datos, construir un sistema capaz de predecir a que clase pertenecen nuevos datos que se introduzcan al modelo. La *SVM* representa en un eje de coordenadas los vectores de entrenamiento, separando las clases por un espacio lo más grande posible, cuando nuevos datos son introducidos se colocan en el eje en el espacio más cercano correspondiente a la clase que pertenece (Cortes, 1995).

- *SVM* caso linealmente separable:

Los casos linealmente separables se caracterizan por tener una sola salida y un número bajo de entradas. En estos casos, cada entrada se encuentra a una distancia considerable de las demás, lo que permite establecer un hiperplano claro de separación entre las clases. Un ejemplo sencillo es la separación de dos clases de acordes: Do Mayor y La Mayor. En este escenario hipotético, cada clase se ubicará en un plano bidimensional con una separación espacial significativa entre ellas. Esto facilita observar el comportamiento de la máquina según las características de la señal de entrada, y las líneas resultantes representarían los posibles espacios lineales de discriminación (Paez, 2014).

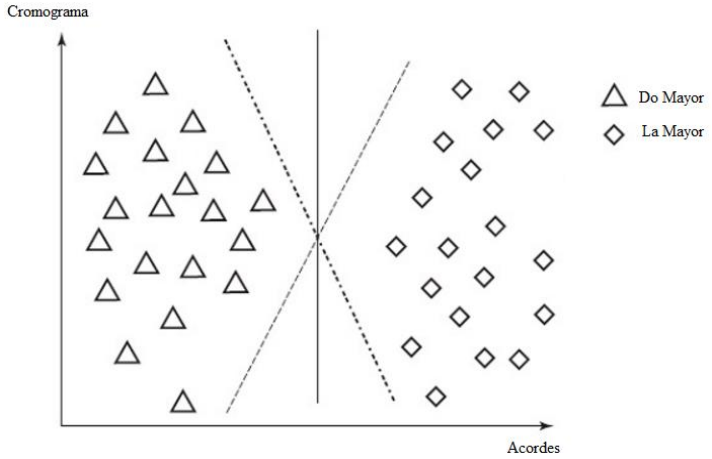

Figura 7. Separación hipotética de dos clases de acordes a partir de una franja lineal, cada una de las líneas representan posibles hiperplanos de separación. Extraído de (Sánchez Anzola, 2015)

Como vemos en la figura 7, hay una infinita posibilidad de hiperplanos con la capacidad de separar las clases en dos grupos, de esta afirmación nace la duda ¿cuál de ellos escoger? La solución es simple, como lo dictaría la lógica, el plano que se encuentre en la posición más alejada tanto de la clase Do mayor como de la clase La mayor, para determinar la distancia máxima se debe tomar en cuenta los datos que se encuentren en los límites de la zona de cada clase, a estos datos se los llamará Vectores de Soporte y el punto intermedio entre la distancia de ellos formará el plano correspondiente a la solución de linealidad del problema

de separación, en la figura 8 se ilustra la selección de un plano según la margen de los vectores de soporte.

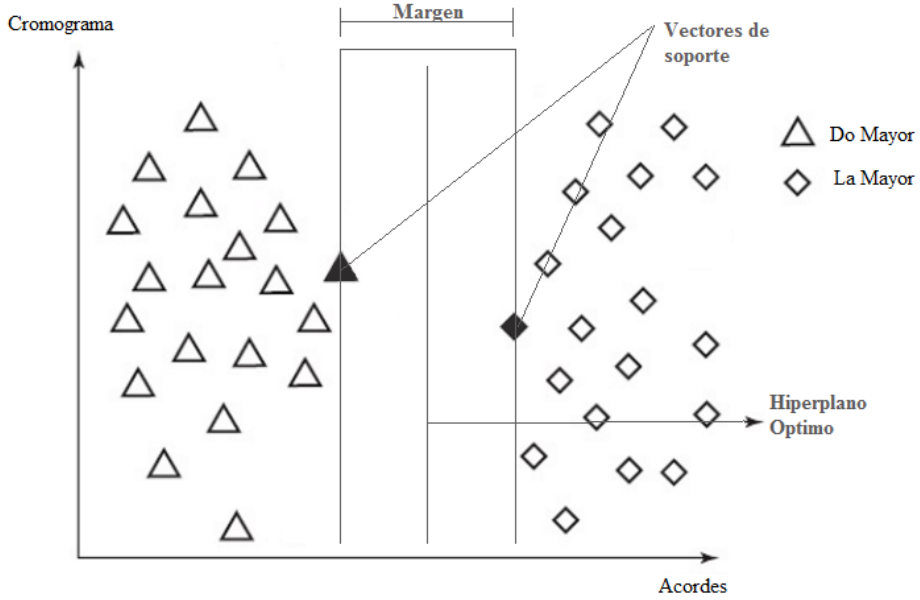

Figura 8. Configuración de hiperplano óptimo a partir de los vectores de soporte. Extraído de (Sánchez Anzola, 2015)

Vapnik propuso formular una relación matemática que maximizara el margen de separación a partir de un determinado conjunto de datos de entrenamiento, para lo cual, se precisó la ejecución de una etapa de aprendizaje donde se realice la búsqueda del hiperplano *h(x)=0* óptimo para la separación del conjunto de datos según la clase $Y \in \{1, -1\}$ a la que pertenecen: $* \, conjunto\ entrenamiento = \{(x_k, y_k) \in \{1, -1\}\} \quad k = 1, \dots, N.$ (Cherkassky, 2004).

Los puntos x que se encuentren en el hiperplano óptimo de separación satisfacen la relación de la ecuación 7.

$$x * w + b = 0 \tag{7}$$

donde:

- w corresponde al vector normal y perpendicular al hiperplano
- x*w corresponde al producto entre los dos vectores

Como la margen es establecida a partir de los datos de entrenamiento, siendo esta la distancia entre las proyecciones perpendiculares de la posición de los vectores de soporte, el conjunto de datos debe cumplir las siguientes restricciones: $x_k * w + b \geq +1 \, para \, y_k = +1$ y $x_k * w + b \geq -1 \, para \, y_k = -1$.

Aplicando cada restricción de la ecuación 7 tenemos finalmente las ecuaciones de los hiperplanos que forman la margen del hiperplano óptimo de separación:

$$H1 = x_k * w + b = 1$$
$$H2 = x_k * w + b = -1 \tag{8}$$

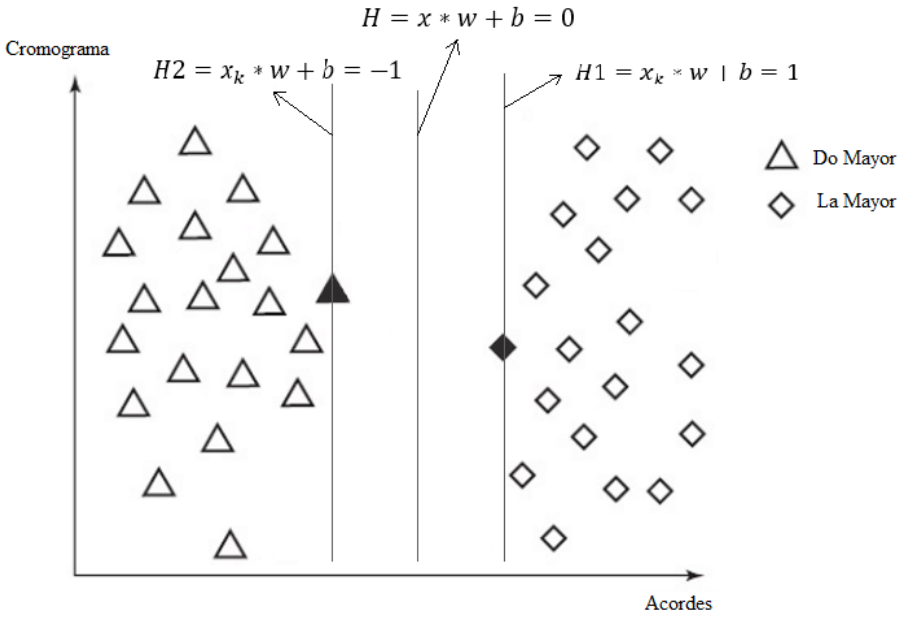

Figura 9. Representación de los hiperplanos de margen y su interacción con el hiperplano óptimo. Extraído de (Sánchez Anzola, 2015)

En la figura 9 se hace una aclaración con respecto a los hiperplanos de margen (*H1* y *H2*) y su relación con el hiperplano óptimo (*H*), se ve claramente como la margen se maximiza a través de los vectores de soporte, puesto que más allá de ellos no existen datos intermedios que cambien el espacio de separación de clases.

- *SVM* caso no linealmente separable empleando *Kernel*

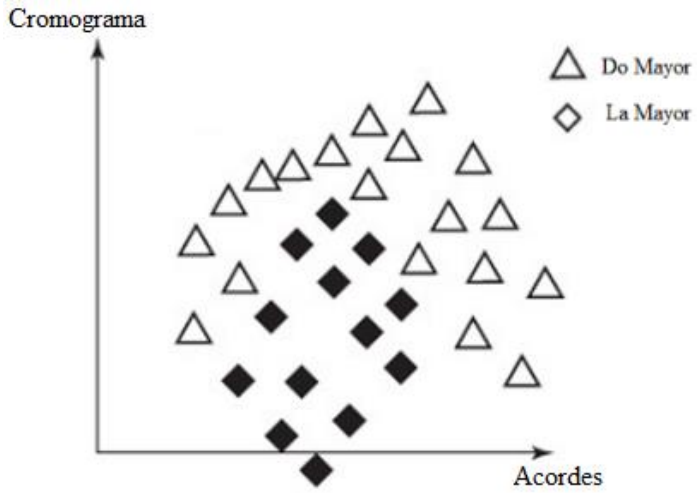

Figura 10. Espacio de entrenamiento con muestras no linealmente separables. Extraído de (Sánchez Anzola, 2015)

El verdadero potencial de las *SVM* se presenta al desarrollar problemas del tipo no lineal, puesto que este es realmente el objetivo para el cual fueron diseñadas, tenemos pues, que las condiciones de margen para el hiperplano en el espacio inicial (no *Kernelizado*) no se cumplen, como se ve en la figura 10 no hay función lineal capaz de separar dichas clases tal

como están dispuestas. Para abordar el caso se emplea un procedimiento denominado *Kernel Trick*, por el cual es posible transformar los datos a un espacio euclidiano de mayor dimensión en donde las características si puedan ser separadas mediante un hiperplano (Sánchez Anzola, 2015).

$$f(x, \alpha b) = \sum_{k \in SV}^{N} y_k \, \alpha_k K(x_k, x) + b \tag{9}$$

Donde:

- y_k Representa a la etiqueta o clase correspondiente al vector de soporte.
- α_k Representa a los multiplicadores de *Lagrange.*
- N Representa la cantidad de vectores de soporte en el espacio.
- k Representa el vector de soporte.
- K Representa la función *Kernel*.
- b Representa al término de sesgo o intercepto que ajusta el hiperplano de separación.

Una vez aplicado el *Kernel* para la transformación del espacio, la solución de separación se puede abordar con un hiperplano como se ha venido estudiando anteriormente, en la figura 11 se presenta la representación de los datos no linealmente separables, su correspondiente transformación a partir del *Kernel* y el hipotético clasificador no lineal necesario para clasificar las muestras.

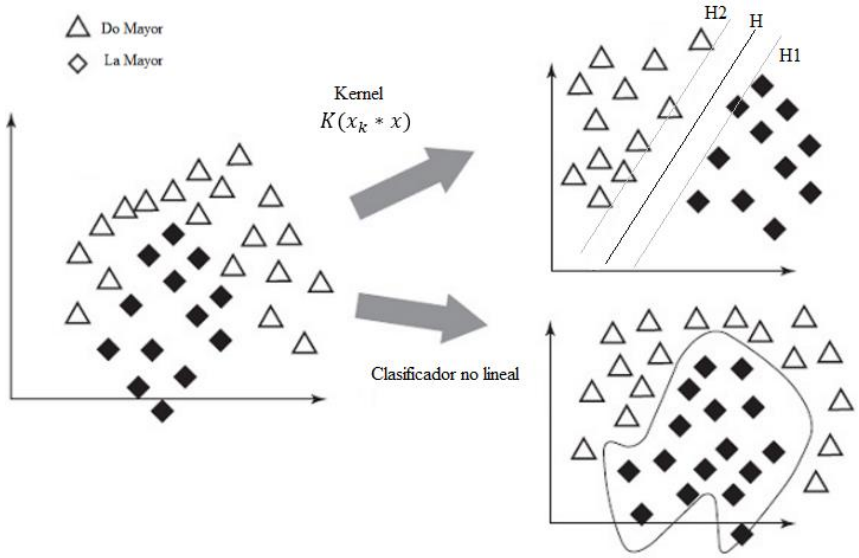

Figura 11. Espacio de datos no lineales con su transformación a través del Kernel y el hipotético clasificador no lineal. Extraído de (Sánchez Anzola, 2015)

2.2.3.3 Arboles de decisión (*Decision Tree*)

Los árboles de decisión son modelos predictivos utilizados tanto en tareas de clasificación como de regresión. Funcionan dividiendo los datos de entrada en subconjuntos más pequeños a través de nodos, que representan decisiones basadas en un conjunto de características. Cada división se realiza con el objetivo de maximizar la pureza de los subconjuntos en términos de la variable objetivo (Kingsford C, 2008).

El algoritmo principal para la construcción de un árbol de decisión es recursivo, y se basa en criterios de selección que permiten elegir la mejor característica (o atributo) que divida el conjunto de datos en cada paso. Para construir un árbol de decisión se requiere el dominio de los siguientes conceptos:

- **Entropía**: mide la impureza de un conjunto de datos en términos de su variabilidad. Para una variable aleatoria X, con posibles valores $x_1, x_2, \dots, x_n$ la entropía se define como:

$$H(X) = -\sum_{i=1}^{n} P(x_i) log_2 P(x_i) \tag{10}$$

  Donde:
  - $P(x_i)$ Es la probabilidad de que el evento x_i ocurra

- **Ganancia de Información**: se utiliza para elegir la característica optima en cada nodo del árbol. Mide la reducción de la entropía después de dividir los datos según una característica. Se define como:

$$IG(Y, X) = H(Y) - H(Y|X) \tag{11}$$

  Donde:
  - X Es la variable predictora o independiente.
  - Y Es la variable objetivo.
  - $IG(Y, X)$ Es la ganancia de información obtenida sobre la variable Y.
  - $H(Y)$ Es la entropía de la variable aleatoria.
  - $H(Y|X)$ Es la entropía condicional de Y dado X.

- **Índice de Gini**: mide la probabilidad de que un elemento seleccionado al azar sea clasificado incorrectamente si se etiqueta de acuerdo con la distribución de clases en el conjunto de datos. Se define como:

$$Gini(X) = 1 - \sum_{i=1}^{n} P(x_i)^2 \tag{12}$$

2.2.3.4 Bosques Aleatorios (*Random Forest*)

Los *Random Forests* (Bosques Aleatorios), propuestos por *Breiman* (Breiman, 2001), son una combinación de predictores basados en árboles de decisión. Cada árbol depende de los valores de un vector aleatorio muestreado de manera independiente y con la misma distribución para todos los árboles en el bosque. A medida que el número de árboles en el bosque crece, el error de generalización converge (casi seguramente) hacia un límite. La eficacia del bosque de clasificadores basados en árboles depende de la fortaleza de los árboles individuales y de la correlación entre ellos. En la figura 13 se presenta el esquema básico de un random forest donde las soluciones convergen en una única salida.

La estrategia de utilizar una selección aleatoria de características para dividir cada nodo produce tasas de error que son comparables favorablemente con *Adaboost* (Saitta, 1996), pero son más robustas frente al ruido. Las estimaciones internas supervisan el error, la fortaleza y la correlación, y se utilizan para mostrar la respuesta al aumentar el número de características utilizadas en la división. Además, estas estimaciones internas se utilizan para medir la importancia de las variables, a continuación presentamos algunos conceptos claves a la hora de trabajar con bosques aleatorios:

- **Aleatorización de datos y variables:** un bosque aleatorio construye múltiples árboles de decisión. Cada árbol es entrenado con una muestra aleatoria con reemplazo (método de *bootstrap*) del conjunto de datos. Además, en cada nodo de un árbol, se selecciona un subconjunto aleatorio de las características para determinar la mejor división. Esta aleatorización reduce la correlación entre los árboles, lo que mejora la generalización del modelo.

- **Votación o promedio:** en problemas de clasificación, cada árbol emite un "voto" por la clase predicha, y la clase final se decide por mayoría de votos. En problemas de regresión, se toma el promedio de las predicciones de los árboles:

$$\hat{y} = \frac{1}{T}\sum_{i=1}^{T} \hat{y}_t \tag{13}$$

  Donde:
  - T es el número de árboles.
  - $\hat{y}_t$ es la predicción del t-ésimo árbol.

- **Cálculo del error del modelo:** el error de un bosque aleatorio depende de dos factores:
  - **Correlación entre los árboles** (ρ): si los árboles son altamente correlacionados, el modelo cometerá errores similares. La aleatorización reduce esta correlación.
  - **Precisión de cada árbol** (err): si cada árbol es altamente preciso, el bosque tendrá un mejor rendimiento.

El error general del bosque aleatorio se puede expresar como:

$$\boldsymbol{Error\ del\ Bosque} = \rho * err$$

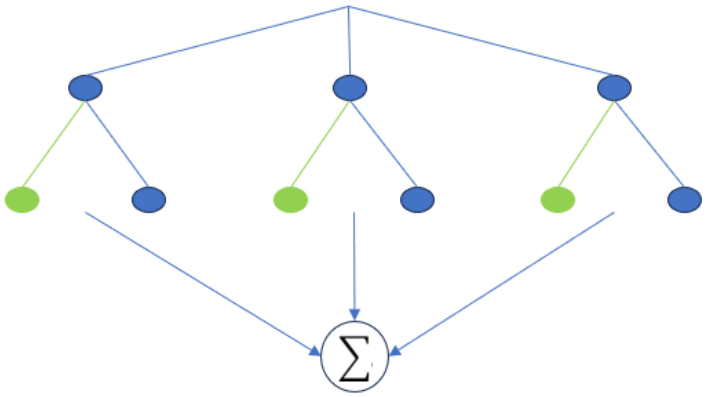

Figura 12. Ejemplo de la construcción del algoritmo random forest. Imagen propia

2.2.3.5 Gradient Boosting Machine

El método *Gradient Boosting Machine* (GBM) puede entenderse como un algoritmo de optimización numérica cuyo objetivo es encontrar un modelo aditivo que minimice la función de pérdida (S. Touzani, 1533–1543). El algoritmo GBM agrega de manera iterativa en cada paso un nuevo árbol de decisiones (también conocido como "aprendiz débil") que mejor reduzca la función de pérdida. En la figura 14 se ejemplifica el funcionamiento del algoritmo.

En el contexto de la regresión, el algoritmo comienza inicializando el modelo con una suposición inicial, que suele ser un árbol de decisiones que minimiza la función de pérdida (en el caso de la regresión, esta función es el error cuadrático medio). Luego, en cada paso, se ajusta un nuevo árbol de decisiones a los residuos actuales y se agrega al modelo anterior para actualizar los residuos. El algoritmo continúa iterando hasta alcanzar un número máximo de iteraciones proporcionado por el usuario. Este proceso se conoce como "paso a paso", lo que significa que en cada nuevo paso, los árboles de decisiones agregados al modelo en pasos anteriores no se modifican. Al ajustar árboles de decisiones a los residuos, el modelo mejora en las regiones donde no se desempeña de manera óptima. A continuación se presenta el paso a paso del funcionamiento del algoritmo

- **Modelo Inicial:** se comienza con un modelo predictivo básico, $F_0(x)$, que puede ser tan simple como predecir el valor medio de la variable objetivo. La función de pérdida a minimizar puede ser el error cuadrático medio (para problemas de regresión) o el logaritmo de la probabilidad (para clasificación).

- **Corrección del error con residuales:** en cada iteración, el algoritmo ajusta un nuevo modelo a los residuales (errores) del modelo anterior. Esto se traduce en ajustar el modelo para predecir la diferencia entre la predicción actual y el valor

real. El nuevo modelo que se ajusta es otro árbol de decisión que trata de minimizar estos residuales.

- **Actualización del modelo:** el modelo se actualiza con una tasa de aprendizaje η para controlar cuánto influye cada nuevo árbol en la predicción final:

$$F_m(x) = F_{m-1}(x) + \eta * h_m(x) \tag{14}$$

Donde:

  - $h_m(x)$ es el nuevo árbol ajustado a los residuales.
  - η es un parámetro de regularización que previene el sobreajuste.
  - m es la etapa o iteración del proceso de boosting.

- **Repetición del proceso:** este proceso de ajuste y actualización se repite hasta que el error deja de disminuir significativamente o se alcanza un número máximo de iteraciones.

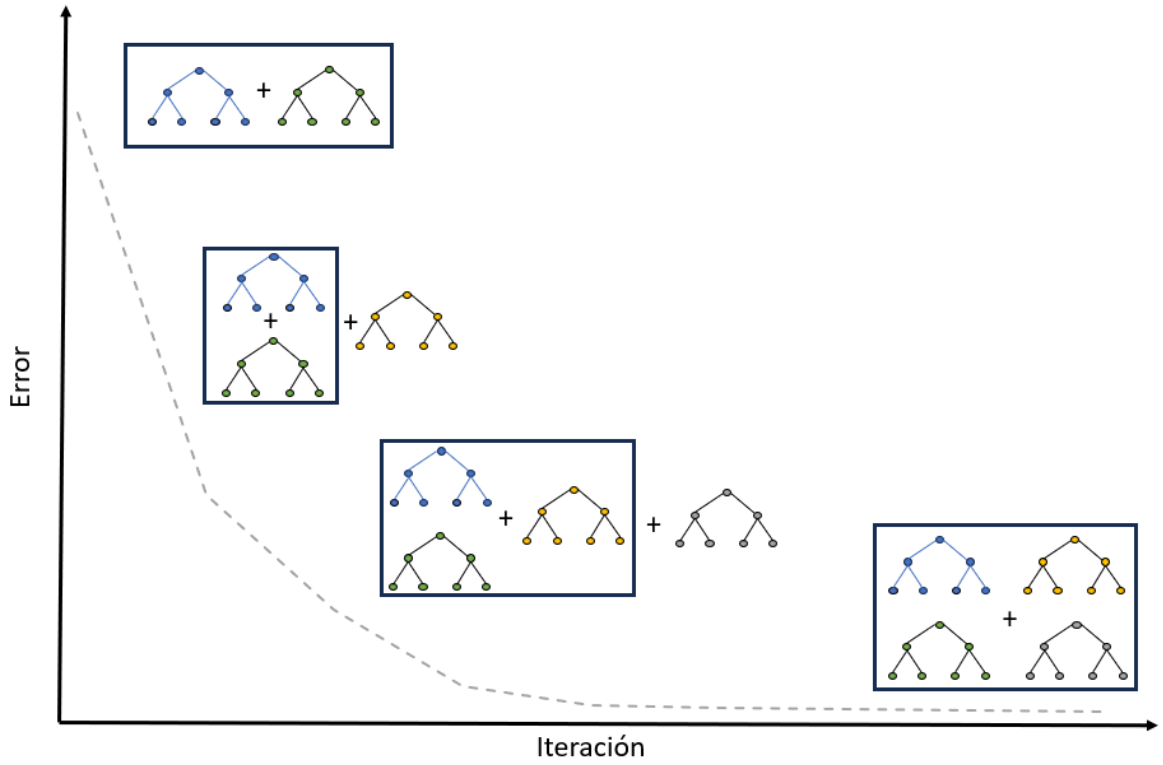

Figura 13. Ejemplo del funcionamiento del algoritmo gradient boosting, a medida en que se adicionan nuevos árboles de decisión a las salidas de los anteriores, se logra disminuir el error de generalización. Imagen propia

2.2.3.6 *Transformers*

Los *Transformers*, introducidos por Vaswani et al. en su artículo "*Attention Is All You Need*" (A. Vaswani, 2017), representan un avance significativo en el campo del Procesamiento de Lenguaje Natural (*NLP*) y aprendizaje automático en general. Esta arquitectura reemplaza los métodos secuenciales de Redes Neuronales Recurrentes (*RNN*) y

Redes Neuronales de Memoria a Largo Plazo (*LSTM*) por un enfoque basado completamente en la atención.

Los *Transformers* son una arquitectura de redes neuronales diseñada para manejar secuencias de datos de una manera más eficiente que los enfoques anteriores, como las *RNN*. Mientras que las *RNN* procesan secuencias de manera recurrente (un dato tras otro), los *Transformers* utilizan un mecanismo de auto-atención que les permite procesar todos los datos simultáneamente. Este paralelismo los hace más rápidos y efectivos en la captura de relaciones a largo plazo entre los elementos de una secuencia.

Los *Transformers* constan de dos partes principales: codificador (*encoder*) y decodificador (*decoder*), cada uno compuesto por múltiples capas. Cada capa de estas partes incluye submódulos de atención y redes *feed-forward*, en la figura 15 se presenta la arquitectura *transformer*. A continuación, se detallan los componentes clave:

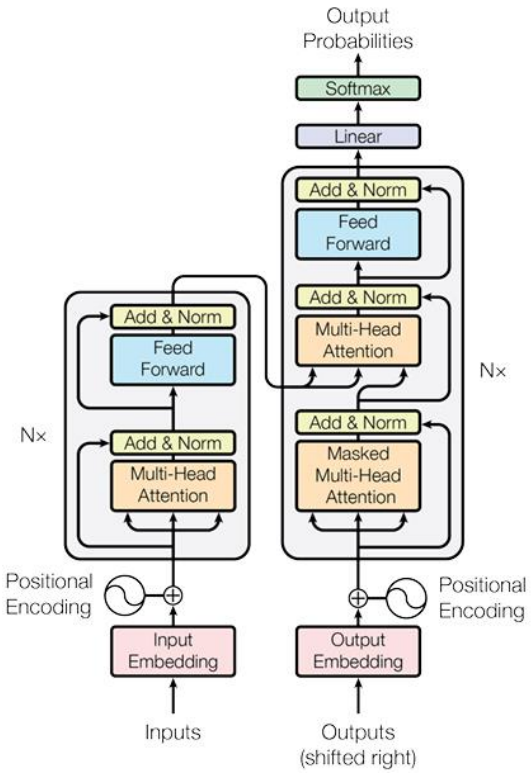

Figura 14. Transformer, arquitectura del modelo. Fuente: (A. Vaswani, 2017)

- Codificador: procesa la secuencia de entrada y genera una representación intermedia que el decodificador utiliza para hacer predicciones. Esta capa tiene dos submódulos:
    - Mecanismo de auto-atención: permite que cada dato de la secuencia de entrada "atienda" a todos los demás datos, calculando qué partes de la secuencia son relevantes.
    - Red *feed-forward*: una red neuronal densa aplicada a cada posición de la secuencia independientemente.

- Decodificador: genera la secuencia de salida, utilizando tanto la representación generada por el codificador como su propio mecanismo de auto-atención para procesar la secuencia de salida generada hasta ese punto. Cada capa del decodificador también tiene dos submódulos de atención: una de auto-atención y otra para la atención cruzada.

- Mecanismo de atención: es la base del modelo *Transformer*. La atención permite al modelo enfocarse en partes relevantes de la secuencia de entrada para cada paso de la secuencia de salida (figura 16, izquierda), en lugar de procesar secuencias completas de manera secuencial como lo hacían los modelos anteriores.

  La atención en los *Transformers* se calcula usando la ecuación 15:

$$Attention(Q, K, V) = softmax\left(\frac{QK^t}{\sqrt{d_k}}\right)V \quad (15)$$

  Donde:
  - Q es la matriz de consultas.
  - K es la matriz de claves.
  - V es la matriz de valores.
  - d_k es la dimensión de los vectores clave, utilizada para normalizar el producto punto entre Q y K.
  -

  Este cálculo permite que cada dato o *token* "atienda" a otros datos en la secuencia de manera dinámica, ponderando su relevancia.

- *Multi-Headed Attention*: en lugar de realizar una única operación de atención, los *Transformers* emplean un mecanismo *llamado multi-headed attention* (figura 16, derecha). Esto permite que diferentes "cabezas" de atención se enfoquen en distintas partes de la secuencia de manera simultánea. Cada cabeza de atención realiza su propio cálculo de atención y los resultados se combinan al final.

  El proceso de multi-cabezas de atención se describe de la siguiente manera:

  1. Se dividen las matrices Q, K y V en varias submatrices más pequeñas.
  2. Cada submatriz pasa por su propio mecanismo de atención.
  3. Los resultados se concatenan y proyectan de nuevo para obtener el valor final de la atención.

  Esto se puede describir como se ve en la ecuación 16:

$$MultiHead(Q, K, V) = Concat\ (head_1, \dots, head_h)W^o \quad (16)$$

  Donde cada $head_1$ es el resultado de una operación de atención, y W^o es una matriz de pesos que proyecta las salidas concatenadas.

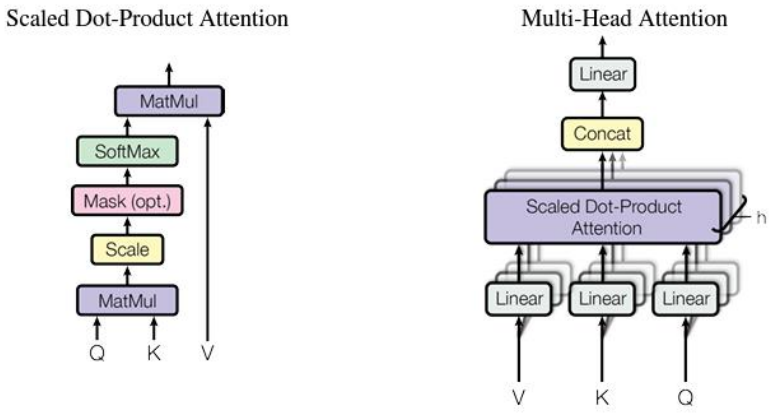

Figura 15. (Izquierda) Mecanismo de atención. (Derecha) Multi-Headed Attention. Fuente: (A. Vaswani, 2017)

- Redes *Feed-Forward*: después de aplicar la atención, cada posición en la secuencia pasa a través de una red *feed-forward* completamente conectada, que consiste en dos capas lineales con una función de activación entre ellas:

$$FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2 \tag{17}$$

  Donde W_1, W_2, b_1 y b_2 son matrices de pesos y sesgos aprendidos. Esta red se aplica de manera independiente a cada posición en la secuencia.

- *Embeddings* posicionales: dado que los *Transformers* no procesan secuencias en orden, como lo hacen las *RNN*, necesitan incorporar la información posicional para saber en qué lugar de la secuencia se encuentra cada palabra. Esto se logra añadiendo *embeddings* posicionales a los vectores de entrada y salida.

  Los *embeddings* posicionales se calculan como funciones trigonométricas, basadas en la posición del *token* en la secuencia:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \tag{18}$$

  Donde:
  - pos es la posición del elemento en la secuencia.
  - i es el índice de la dimensión del vector de codificación posicional.
  - d es la dimensionalidad del vector de la representación de la secuencia.

2.2.3.7 *VGGish*

VGGish es una red neuronal preentrenada diseñada para extraer representaciones compactas y significativas de señales de audio. Este modelo se basa en la arquitectura de la red *Visual Geometry Group* (VGG), originalmente desarrollada para la clasificación de imágenes, pero adaptada para el procesamiento de audio (Castro-Ospina, 2024).

VGGish procesa el audio dividiéndolo en fragmentos de 0.96 segundos, y luego calcula un espectrograma log-Mel de cada fragmento, que sirve como entrada para la red neuronal. Los espectrogramas Mel son representaciones en frecuencia-temporal que capturan las características relevantes de las señales de audio.

La arquitectura está compuesta por varias capas convolucionales y de *max-pooling*. Las capas convolucionales aplican filtros a los espectrogramas de entrada para detectar patrones locales y extraer características de bajo nivel. Posteriormente, las capas de *max-pooling* reducen las dimensiones de los mapas de características obtenidos, conservando la información más importante. Esto permite que el modelo capture patrones relevantes a diferentes escalas.

Finalmente, las capas totalmente conectadas de la red toman las salidas planas de las capas anteriores y las mapean a una representación de 128 dimensiones. Este vector de características encapsula la información importante del audio y es utilizado en diversas tareas, como la clasificación de audio, la recuperación basada en contenido y el análisis de escenas acústicas.

2.2.3.8 *HuBERT*

HuBERT (*Hidden-Unit BERT*) es un modelo de aprendizaje auto-supervisado que tiene como objetivo aprender representaciones del habla a partir de datos sin etiquetar mediante la predicción de unidades acústicas ocultas. Inspirado por los avances en modelos de lenguaje como BERT (J. Devlin, 2018), *HuBERT* adapta su metodología de predicción enmascarada al dominio del habla, buscando capturar tanto patrones acústicos locales como dependencias temporales de largo alcance.

El aprendizaje auto-supervisado ha emergido como una alternativa a los métodos supervisados que dependen de grandes cantidades de datos etiquetados. En el contexto del procesamiento del habla, anotar grandes volúmenes de datos con transcripciones precisas es costoso y laborioso. *HuBERT* resuelve esta limitación entrenando en un gran conjunto de datos no etiquetados, donde el modelo aprende a predecir los patrones ocultos en el audio de manera automática, sin supervisión humana directa (W.-N. Hsu, 2021).

HuBERT sigue la estructura de un modelo *Masked Language Model* (MLM), pero aplicado al dominio del audio (figura 17). La predicción enmascarada se realiza no sobre palabras o caracteres, sino sobre representaciones acústicas. A continuación se presenta el modo de operación de *HuBERT*:

- Extracción de características iniciales: en la primera fase, las señales de audio de entrada se dividen en segmentos de igual longitud, y se extraen características acústicas como los Coeficientes Cepstrales de Frecuencia de Mel (*MFCCs*). Estas características se utilizan como representaciones del habla en lugar de las señales crudas.

- *Clustering* y pseudo-etiquetas: para guiar el aprendizaje, *HuBERT* utiliza una técnica de clustering para agrupar los segmentos de audio en *unidades acústicas latentes*. Estas unidades latentes no son más que agrupaciones de las características acústicas, generadas mediante métodos de clustering no supervisado, como el algoritmo *K-means*. Estas pseudo-etiquetas, generadas de forma automática, se utilizan como el objetivo a predecir durante el entrenamiento del modelo.

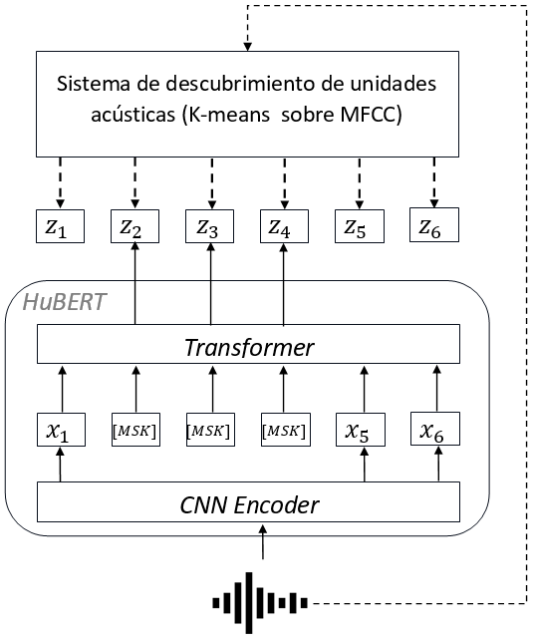

Figura 16. El enfoque de HuBERT predice asignaciones de clúster ocultos para los cuadros enmascarados generados por una o varias iteraciones de agrupamiento k-means. Imagen propia

- Predicción de unidades ocultas: en cada iteración de entrenamiento, una porción de las unidades acústicas de los segmentos de entrada es enmascarada. La tarea del modelo es predecir estas unidades enmascaradas utilizando el contexto circundante de los segmentos no enmascarados.

  Matemáticamente, el modelo minimiza una función de pérdida como se ve en la ecuación 18:

$$\mathcal{L} = -\sum_{t \in M} logP(u_t|X;\theta) \tag{19}$$

  Donde:
  - M es el conjunto de índices correspondientes a las unidades enmascaradas.

- $P(u_t|X;\theta)$ es la probabilidad de predecir correctamente la unidad acústica u_t en el tiempo t, da la secuencia de la entrada X y los parámetros del modelo θ

Este enfoque se inspira directamente en el mecanismo de predicción enmascarada de *BERT*, donde el modelo aprende representaciones útiles al tratar de inferir las partes faltantes del *input*.

- Ajuste del modelo: el entrenamiento de *HuBERT* es progresivo, lo que significa que una vez que el modelo ha aprendido a predecir un conjunto de unidades latentes, estas se utilizan para actualizar el conjunto de pseudo-etiquetas y mejorar aún más el entrenamiento. Este ciclo de retroalimentación permite que el modelo mejore iterativamente su capacidad para representar las características acústicas y temporales del habla.

2.2.3.9 Ajuste fino (*Fine-Tunning*)

Fine-tuning, también conocido como ajuste fino, constituye una práctica extendida en el ámbito del aprendizaje automático y la inteligencia artificial, especialmente en lo que respecta al entrenamiento de modelos de redes neuronales previamente entrenados. Su esencia radica en la adaptación de un modelo pre-entrenado en una tarea específica para su aplicación en una tarea relacionada o específica (A. Radford, 2018).

El procedimiento de *fine-tuning* implica tomar un modelo previamente entrenado en un conjunto de datos amplio y general y establecer sus parámetros como punto de partida para realizar el entrenamiento con nuevos datos. Radfor en (A. Radford, 2018) presenta esta metodología como un método para encontrar un buen punto de inicialización en lugar de desarrollar los modelos desde cero. Su investigación ha demostrado que el preentrenamiento actúa como un esquema de regularización, permitiendo una mejor generalización en redes neuronales profundas. Este método se utiliza ampliamente para ayudar a entrenar redes neuronales profundas en varias tareas como clasificación de imágenes, reconocimiento de voz, desambiguación de entidades y traducción automática (J. Devlin, 2018).

Usualmente, el ajuste fino se lleva a cabo con una tasa de aprendizaje menor que la utilizada en el entrenamiento original. Esto permite que el modelo se adapte gradualmente a los nuevos datos, evitando olvidar el conocimiento adquirido durante el entrenamiento previo.

El *fine-tuning* es una técnica sumamente poderosa, pues posibilita aprovechar el conocimiento preexistente en modelos pre-entrenados, lo que conlleva frecuentemente a un mejor rendimiento y a una convergencia más rápida en tareas específicas, sobre todo cuando se dispone de conjuntos de datos de entrenamiento limitados.

3. COMPRENSIÓN DE LOS DATOS

Antes de presentar cada uno de los pasos que se realizaron para desarrollar el detector automático de acordes, se debe tener en cuenta que en este trabajo empleamos dos bases de acordes diferentes. La primera es un banco llamado '*AudioPianoTriadDataset'*, que consta de 42.300 muestras de audio y es una base que se encuentra disponible en: https://zenodo.org/records/5217057, la segunda es una base creada para este trabajo desde 0 en un secuenciador digital denominado *FLstudio*, la cual tiene un total de 960 muestras de audio y se diferencia de la anterior en que no contiene muestras de acordes alterados como lo son los aumentados o disminuidos.

3.1 Banco de acordes '*AudioPianoTriadDataset*'

El conjunto de datos *'AudioPianoTriadDataset'*, que consta de 42.300 muestras de audio. Este conjunto de datos presenta las siguientes características principales: contiene 300 audios por tipo de acorde, interpretados en un piano digital en tres octavas diferentes (3, 4, 5), con tres niveles de volumen distintos (piano, mezzoforte, forte), y en cuatro modos diferentes (mayor, menor, aumentado, disminuido). Cada audio se grabó con una frecuencia de muestreo de 16 kHz, una duración de 4 segundos y una calidad de 16 bits PCM, lo que resulta en archivos individuales de 128 kB. Además, es importante destacar que el conjunto de datos está balanceado, es decir, contiene la misma cantidad de muestras de audio para cada etiqueta, de igual forma, se debe tener en cuenta la homogeneidad en la generación de las muestras y el ambiente controlado bajo el cual fueron obtenidas. Cada muestra fue grabada utilizando un piano digital, sin ningún tipo de ruido ambiental o la inclusión de otros instrumentos que puedan distorsionar la señal del acorde, lo que eventualmente puede constituir una falta de características ingresadas a los modelos de tal forma que los resultados no sean los esperados al emplearlos en condiciones de interpretación como lo es una pieza musical.

3.2 Banco de acordes con *FLstudio*

Dado que el conjunto de datos presentado anteriormente tiene características muy similares entre clase, es decir, dentro de una misma clase los audios son muy similares, además, que presentan una falta de elementos de interpretación musical, se decidió construir un set extra de grabaciones que permitieran contrastar los resultados obtenidos, dado que este nuevo set presenta características semejantes al sonido de un acorde en medio de una pieza musical donde se incluyen más instrumentos o incluso percusiones. Para el desarrollo de este proyecto se utilizó el *software FLstudio*, de la marca *Image-Line*, una estación de trabajo de audio digital que funciona como editor y secuenciador multipista con soporte MIDI, ampliamente utilizado en la producción musical. A través de esta herramienta, se generaron tres bancos de acordes, como se muestra en la figura 18. Los dos primeros bancos contienen 240 muestras de acordes, mientras que el tercero incluye 480, con una duración aproximada de cuatro segundos por acorde. La principal diferencia entre estos bancos radica en el número de instrumentos utilizados. El primer banco recrea los acordes empleando solo un piano o guitarra (la mitad de los acordes con piano y la otra mitad con

guitarra). El segundo banco añade percusión al piano/guitarra, y el tercero incorpora, además de la percusión, un bajo, tal como se ilustra en la figura 20. Los bancos de 240 acordes presentan 10 acordes de cada denominación (C, Cm, …, Bm) los 5 primeros presentan como instrumento armónico el piano, los otros 5 presentan como instrumento armónico la guitarra y en dos modos diferentes (mayor y menor).

Cada audio se grabó con una frecuencia de muestreo de 48 kHz, una duración de 4 segundos y una calidad de 16 bits PCM, lo que resulta en archivos individuales de 225 kB.

Figura 17. Creación banco de acordes con FLstudio. Imagen propia

4. PREPARACIÓN DE LOS DATOS

4.1 Etiquetado de los datos

En el proceso de entrenamiento de modelos supervisados, resulta fundamental contar con un conjunto de datos debidamente etiquetado. Para facilitar la preparación de conjuntos de datos, incluyendo aquellos que involucran datos de audio, *Hugging Face* ofrece la librería Datasets. Esta librería permite preparar los datos a partir del sistema de organización en carpetas en las que estén almacenados. Para su uso, se requieren dos carpetas principales denominadas *'train'* y *'test'*. Dentro de estas carpetas principales, es posible crear subcarpetas que correspondan a las etiquetas que se asignarán a los datos contenidos en ellas.

La Figura 19 ilustra cómo se estructuró el sistema de carpetas para las 48 clases de acordes del dataset "*AudioPianoTriadDataSet*". Para el dataset desarrollado con *FLstudio* se empleó la misma estrategia solo que se generaron 24 clases dado la menor cantidad de acordes contenidos donde solo se incluyen los modos mayor y menor.

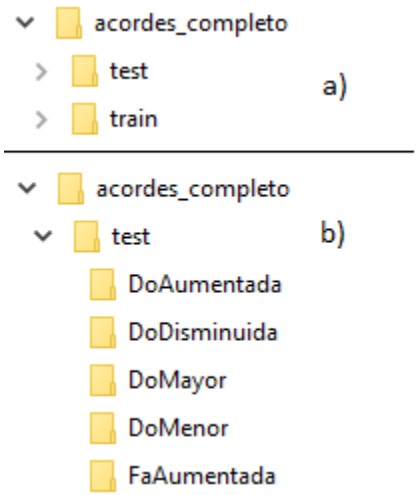

Figura 18. Organización de archivos para la preparación de datos: a) Creación de una carpeta que contiene las particiones de entrenamiento y prueba. b) Dentro de la carpeta de prueba, se encuentran las subcarpetas que representan las clases de los grupos de datos. Imagen propia

4.2 Método de entrenamiento y comprobación

Como pudimos ver en la sección 2.2.3 las máquinas de aprendizaje responden a los estímulos o entrenadas al modelo, es decir, aprenden a partir de un conjunto de datos y son capaces de predecir, con base en su algoritmo de ajuste, que tipo de respuesta debe tener a un nuevo estímulo que desconoce pero que tiene relación con los que ya ha procesado anteriormente. Debido a este comportamiento, es necesario adecuar el conjunto de datos para poder realizar un entrenamiento y una previa comprobación del funcionamiento.

Bradley Efron (Efron, 1995) presenta una forma de tratamiento de estos datos la cual se denomina validación cruzada o *cross validation*, la cual es una práctica estadística que

consiste en fragmentar una muestra de datos en subconjuntos para analizar uno de ellos y luego validar dicho análisis con el resto de los subconjuntos. En diversas bibliografías se presenta una relación 70/30 la cual consiste en realizar el entrenamiento con el 70% de la base de datos (*train*) y realizar la comprobación con el restante 30% (*test*).

Es importante resaltar, que para realizar el entrenamiento de los modelos solo se empleó la base de datos "*AudioPianoTriadDataSet*", el set desarrollado en *FLstudio* fue generado exclusivamente para comprobar la capacidad de los modelos de hacer clasificaciones en condiciones de mayor cantidad de características dentro del audio.

4.3 Extracción de características

Teniendo en cuenta la cantidad de muestras de audio por acorde, fue necesario emplear un dataloader (Bai, 2541-2556) la cual es una herramienta o módulo que facilita la gestión y carga eficiente de los datos durante el desarrollo de proyectos analíticos. Este es un paso importante dentro del desarrollo puesto que de no usarse los recursos de las herramientas computacionales empleadas tienden a desbordarse.

Una vez definido el método de procesamiento, se debe tener en cuenta que se tienen dos métodos de extracción de características, el primero emplea la arquitectura *HuBERT* y requiere del uso de la librería *AutoFeatureExtractor* de *HuggingFace*, el segundo es el modelo *VGGish* para caracterización de audio. Cada uno de ellos generó un archivo *Pickle* con el que se realizó el entrenamiento de los modelos. El formato *pickle* permite serializar y deserializar objetos en *Python*, lo cual permite guardar estructuras de datos, como modelos entrenados o *dataframes*, sin la necesidad de procesarlos nuevamente, por lo que es de gran ayuda dado el volumen de datos trabajados durante el desarrollo del DAA.

4.3.1 Extracción de características con *HuBERT*

La extracción de características se llevó a cabo utilizando la herramienta AutoFeatureExtractor disponible en la plataforma *HuggingFace*. A continuación se presentan los puntos significativos en el proceso de caracterización de audios llevado a cabo por este método:

- **Carga del modelo preentrenado**: se utilizó el modelo *HuBERT* disponible en *HuggingFace*, el cual ya ha sido preentrenado en grandes cantidades de datos de audio no etiquetados. El modelo fue cargado junto con su correspondiente AutoFeatureExtractor, que permite extraer las características de cada archivo de audio.

- **Extracción de características**: a partir de cada archivo de audio de entrada, *HuBERT* genera un conjunto de representaciones de características que son esencialmente embeddings de audio. Estos *embeddings* se extraen en forma de un tensor multidimensional que captura la información temporal y frecuencial del audio. En particular, para cada segundo de audio, el modelo *HuBERT* genera un

total de 768 características por trama, lo que proporciona una representación de alta dimensionalidad capaz de capturar detalles precisos sobre la estructura del sonido.

- **Cantidad de características por archivo**: El número de tramas o "*frames*" generados para un archivo de audio depende de la duración de este, ya que *HuBERT* realiza una segmentación temporal a una tasa fija de 50 tramas por segundo**.** En nuestro caso, los archivos de audio empleados son de 4 segundos lo que genera aproximadamente 200 tramas, resultando en un total de 153,600 características. Este alto nivel de granularidad permite que el modelo capture tanto las variaciones sutiles como las más evidentes en las señales acústicas.

- **Interpretación de las características**: las características extraídas por *HuBERT* son representaciones latentes que encapsulan información esencial sobre la señal de audio, incluyendo el timbre, la tonalidad y las variaciones en la intensidad del sonido a lo largo del tiempo. Estas representaciones no son características interpretables directamente por humanos, pero son de vital importancia para que los modelos de *machine learning* puedan identificar patrones inherentes en los acordes musicales. Las características extraídas permiten al modelo distinguir entre diferentes clases de acordes, ya que capturan los cambios de frecuencia y armónicos que definen cada acorde.

El resultado fue guardado a través de checkpoints a dos archivos generales llamados *train_features.pkl* y *test_features.pkl.*

4.3.2 Extracción de características con *VGGish*

Para el proceso de reconocimiento de acordes en este trabajo, también se empleó la arquitectura *VGGish* explicada en la sección 2.2.3.7. A continuación se presentan los puntos importantes a la hora de caracterizar un audio con esta herramienta.

- **Carga del modelo preentrenado**: el modelo debe ser cargado a partir de sus librerías en *Python*. Este modelo fue preentrenado en el conjunto de datos *AudioSet* y está diseñado para procesar secuencias de audio a través de una serie de convoluciones, generando *embeddings* que capturan las características clave de las señales de audio.

- **Segmentación del audio**: *VGGish* requiere que los audios sean segmentados en fragmentos de 0.96 segundos para su procesamiento. Dado que los audios utilizados en este trabajo tienen una duración de 4 segundos, cada archivo fue dividido en aproximadamente 4 segmentos, asegurando que cada uno de ellos sea transformado en una representación de alta dimensionalidad que capture tanto la estructura temporal como frecuencial del sonido.

- **Extracción de características**: por cada segmento de 0.96 segundos, *VGGish* genera un vector de características o *embedding* de tamaño 128. Este vector

encapsula la información acústica relevante, como los patrones de frecuencia y las variaciones en la intensidad del sonido, lo cual es esencial para describir el contenido del audio. Estas características no son directamente interpretables por humanos, pero proporcionan a los modelos de *machine learning* la capacidad de identificar patrones complejos que pueden estar asociados a diferentes clases de acordes.

- **Cantidad de características por archivo**: dado que cada archivo de audio de 4 segundos se divide en aproximadamente 4 segmentos, *VGGish* genera un total de 4 vectores de 128 características, es decir, 512 características por archivo de audio (4 segmentos × 128 características por segmento). Esta estructura captura las variaciones acústicas dentro de los 4 segundos de audio, proporcionando un nivel de granularidad suficiente para representar con precisión los acordes musicales.

- **Promedio de los segmentos**: para simplificar el uso de estas características en los modelos de aprendizaje automático, fue necesario realizar un promedio de los 4 vectores de características extraídos de cada archivo. Esto se realizó calculando el promedio de los valores en cada una de las 128 dimensiones de los vectores correspondientes a los 4 segmentos. El resultado es una única representación de 128 características por archivo de audio, lo cual encapsula toda la información relevante del archivo en un solo vector.

El resultado fue guardado a través de checkpoints a dos archivos generales llamados train*_featuresvg.pkl* y test*_featuresvg.pkl.*

Empleando estas dos técnicas de caracterización de audios se procesaron los datos *train/test* de la base "*AudioPianoTriadDataSet*" así como los generados con *FLstudio*, quedando listos los vectores de características para el entrenamiento y la validación de los modelos analíticos.

5. MODELADO

Una vez obtenidos los vectores de características para el entrenamiento y de comprobación, se programaron diferentes scripts en python que permiten realizar el entrenamiento de las 4 máquinas de aprendizaje propuestas en el trabajo, la realización de dicha tarea se presenta en los apartados 5.1 y 5.2.

Se debe tener en cuenta que durante el entrenamiento de los modelos se fraccionó el conjunto de datos de train en un 80/20 con el fin de agregar una etapa de validación anterior a la comprobación con los datos de *test*.

5.1 Modelo basado en arquitectura *HuBERT*

Para utilizar la arquitectura *HuBERT*, es necesario realizar un ajuste fino (fine-tuning) del modelo preentrenado para adaptarlo a las condiciones de los datos que se desean estimar. Al igual que con la herramienta de extracción de características, *HuggingFace* ofrece diversas clases que se utilizan para establecer las condiciones de afinamiento. A continuación, se detallan los pasos para llevar a cabo este procedimiento:

- **Asignación de la etiqueta:** para continuar el proceso de etiquetado del modelo, se debe utilizar la clase *AutoModelForAudioClassification*, la cual asigna automáticamente las etiquetas reconocibles para la arquitectura Distil*HuBERT*.

- **Parámetros de entrenamiento:** utilizando la clase *TrainingArguments*, se definen el tamaño del lote, la acumulación del gradiente, la cantidad de épocas de entrenamiento y la tasa de aprendizaje, en la Tabla 2 se presentan todos los parámetros de entrenamiento empleados. Además, se asigna un nombre al modelo de modo que pueda ser recuperado desde el repositorio de *Hugging Face*.

Tabla 2. Parámetros de ajuste DistillHuBERT

Modelo	***DistillHuBERT***
evaluation_strategy	*epoch*
save_strategy	*epoch*
learning_rate	5.00E-05
per_device_train_batch_size	8
gradient_accumulation_steps	1
per_device_eval_barch_size	8
num_train_epochs	10
warmup_ratio	0.1
logging_steps	5

- **Definición de métricas:** utilizando la librería *'evaluate'*, se debe crear la función de evaluación de la métrica, la cual registrará el progreso del modelo durante las diferentes épocas de entrenamiento.

El modelo se entrenó con el nombre *'alejogil35/distilhubert-finetuned-chorddetection2'* y puede ser utilizado llamándolo desde las diversas clases de la librería de *Transformers*. En el repositorio de *HuggingFace*, se encuentran disponibles los parámetros de aprendizaje y las métricas de entrenamiento y validación de cada época.

5.2 Máquina de Vectores de Soporte (*SVM*)

La Máquina de Vectores de Soporte (*SVM*) se implementó utilizando las librerías de *Python*, específicamente *scikit-learn*, que proporciona una interfaz para configurar y entrenar este tipo de modelos a partir de hiperparámetros ajustables. Es crucial asegurarse de que la matriz de características extraídas cumpla con los requisitos esperados por el modelo, ya que cualquier inconsistencia en los datos podría generar errores durante el proceso de entrenamiento y evitar que el modelo funcione correctamente.

Una *SVM* busca encontrar un hiperplano que maximice la separación entre las clases de los datos. En este caso, las entradas son los vectores de características obtenidos previamente de los audios, y las etiquetas de salida corresponden a las clases de acordes. Para definir el comportamiento del modelo, se deben configurar dos hiperparámetros clave:

- **Parámetro C**: este hiperparámetro controla la tolerancia del modelo a los errores de clasificación. Un valor bajo de C permite márgenes más amplios entre las clases, lo que puede resultar en un modelo más generalizable pero menos preciso. En cambio, un valor alto de C busca clasificar correctamente todos los puntos del conjunto de entrenamiento, lo que puede conducir a un sobreajuste.

- ***Kernel***: define cómo los datos se proyectan en un espacio dimensional superior. Los *Kernel*s más utilizados incluyen el lineal, que busca separar las clases con un hiperplano simple, el *RBF* (*Radial Basis Function*), que proyecta los datos de manera no lineal para manejar problemas más complejos, y el polinomial, que aplica funciones de grado n para separar las clases.

Para optimizar el rendimiento del modelo, se empleó una búsqueda sistemática de los mejores hiperparámetros utilizando la técnica de *GridSearch*. Esta técnica permite explorar diferentes combinaciones de valores para los hiperparámetros C **y** *Kernel*, para nuestra implementación empleamos los parámetros descritos en la Tabla 3, identificando cuál de estas combinaciones produce los mejores resultados en términos de precisión del modelo.

Tabla 3. Parámetros de ajuste SVM con GridSearch

Modelo	***SVM***
Kernel	Lineal, Polinómico, Radial
C	[0.001, 0.01, 0.1, 1, 10, 100]

5.3 Bosque Aleatorio (*Random Forest*)

El *Random Forest* es un modelo que combina múltiples árboles de decisión, donde cada árbol es entrenado a partir de un subconjunto diferente de los datos y características. Esta técnica de agrupación (o *ensemble*) tiene la ventaja de reducir la varianza del modelo, lo que resulta en una mayor robustez y generalización en comparación con un solo árbol de decisión.

Esta técnica de aprendizaje supervisado se encuentra implementada en la librería *scikit-learn*, que permite configurar y entrenar el modelo mediante la manipulación de sus hiperparámetros. Los datos de entrenamiento se entregaron al modelo de la misma forma que para la *SVM*, de igual forma, se empleó la herramienta *GridSearch*. En la Tabla 4 se pueden observar las variaciones de los hiperparámetros empleados en este modelo, con el fin de encontrar los hiperparámetros que optimicen el rendimiento del modelo.

A continuación se presentan los hiperparámetros claves del modelo bosque aleatorio:

- **Número de árboles (*n_estimators)*:** este hiperparámetro define cuántos árboles de decisión se construirán dentro del bosque. Un número mayor de árboles suele mejorar la precisión del modelo, pero a costa de un mayor tiempo de entrenamiento.

- **Máxima profundidad de los árboles (*max_depth*)**: controla cuán profundos pueden ser los árboles de decisión individuales. Un valor más alto permite que los árboles se ajusten más a los datos de entrenamiento, pero podría llevar a sobreajuste.

- **Número mínimo de muestras para dividir un nodo (*min_samples_split*)**: este hiperparámetro define el número mínimo de muestras requeridas para dividir un nodo interno. Un valor más alto previene que los árboles crezcan demasiado profundo, controlando el sobreajuste

- **Número mínimo de muestras en una hoja (*min_samples_leaf*)**: controla el número mínimo de muestras que deben estar presentes en una hoja (el nodo final de un árbol). Este valor evita que se creen hojas con muy pocas muestras, lo que podría inducir un modelo con alta varianza.

- **Número máximo de características a considerar por división (*max_features*)**: este parámetro define cuántas características se consideran en cada división del árbol. Reducir el número de características seleccionadas puede aumentar la diversidad entre los árboles, mejorando la capacidad del modelo para generalizar.

Tabla 4. Parámetros de ajuste Random Forest con GridSearch

Modelo	**Random Forest**
n_estimators	[10, 50, 100]
max_depth	[None, 10, 20, 30]
min_samples_split	[2, 5, 10]
min_samples_leaf	[1, 2, 4]
max_features	[*'auto'*, *'sqrt'*, 'log2']

5.4 *Gradient Boosting Machine*

El *Gradient Boosting* es una técnica de aprendizaje supervisado que crea un modelo robusto a partir de la combinación secuencial de varios modelos débiles, como árboles de decisión, donde cada modelo trata de corregir los errores del anterior. Para su implementación empleamos la librería *XGBoost* la cual es una implementación optimizada del algoritmo de *boosting* que permite tener un mayor rendimiento a la hora del procesamiento de las características.

Los principales hiperparámetros que determinan el comportamiento y la efectividad de este modelo son:

- **Número de estimadores (*n_estimators*)**: este hiperparámetro controla cuántos árboles se crearán en el modelo final. Un número mayor de árboles generalmente mejora la capacidad del modelo para ajustarse a los datos, pero también puede aumentar el riesgo de sobreajuste y el tiempo de entrenamiento.

- **Profundidad máxima de los árboles (*max_depth*)**: limita la profundidad de cada árbol de decisión dentro del modelo. La profundidad controla hasta qué nivel los árboles pueden dividirse en función de los datos. Un valor mayor de *max_depth* permite a los árboles ajustarse mejor a los datos de entrenamiento, pero también puede aumentar la varianza y provocar sobreajuste si no se controla adecuadamente.

- **Tasa de aprendizaje (*learning_rate*)**: define el peso o la importancia que cada nuevo árbol tiene en el modelo combinado. Un valor bajo de *learning_rate* hace que el modelo avance más lentamente, lo que puede mejorar la capacidad de generalización, aunque requiere un mayor número de árboles (*n_estimators*). Por el contrario, una tasa de aprendizaje alta acelera el entrenamiento, pero puede llevar a sobreajuste.

Al igual que con la *SVM* y los *Random Forest*, para encontrar los hiperparámetros adecuados para el modelo, se empleó el método *GridSearch* con los datos que se encuentran en la Tabla 5.

Tabla 5. Parámetros de ajuste Gradient Boosting con GridSearch

Modelo	***Gradient Boosting***
n_estimators	[50, 100, 200]
max_depth	[3, 4, 5]
learning_rate	[0.1, 0.01, 0.001]

6. EVALUACIÓN

Las pruebas realizadas se dividieron en dos partes, correspondientes a los dos conjuntos de datos descritos en la sección 3. En la primera parte, se evaluó la capacidad de los modelos para estimar las clases de acordes utilizando exclusivamente el conjunto de datos "*AudioPianoTriadDataSet*". Cabe destacar que estas grabaciones presentan características muy similares entre clases, ya que las variaciones se limitan al volumen, la intensidad o la octava en la que se generan. Sin embargo, no incluyen otros elementos presentes en composiciones musicales más complejas, como bajos, guitarras o percusión que suenan simultáneamente con la ejecución del acorde. Para la segunda parte, se empleará el set de datos creados en *FLstudio* el cual cuenta con acordes interpretados con diferentes propiedades, desde un piano en solitario hasta bajo y percusión de fondo para el instrumento armónico.

Para los modelos clásicos, además de los dos conjuntos de datos, se evalúan también dos métodos de extracción de características, el primero basado en *HuBERT* y el segundo en *VGGish* tal como se explicó en la sección 4.3.

A continuación se presentan los dos escenarios para cada uno de los modelos entrenados.

6.1 Resultados *HuBERT*

Durante el entrenamiento del modelo *HuBERT*, se pudo evidenciar una pronta adaptación a los datos, el fine tunning del modelo empleando *GPU* logró procesar los datos en menos de 3 horas, en la figura 21 se presenta el *Training Loss*, donde podemos observar al inicio, entre los 0 y los 1000 pasos, que la perdida es alta, en torno a 4, esto es común ya que el modelo aún está en la fase inicial de aprendizaje y no ha ajustado bien sus parámetros. Entre los 1000 y 4000 pasos, se observa una rápida disminución de la perdida. Esto indica que el modelo logra aprender rápidamente a partir de los datos de entrenamiento, lo que infiere que el optimizador está ajustando bien los pesos para reducir los errores. Entre los 4000 y 6000 pasos, la perdida se estabiliza y alcanza valores cercanos a 0. En esta fase, el modelo alcanza una convergencia, e indica que los ajustes adicionales a los parámetros no generan mejoras significativas. Entre los 6000 y 9000 pasos continúa la tendencia estable de la perdida de entrenamiento aunque se ven algunas oscilaciones que finalmente no generan mayores complicaciones de adaptación por lo que se puede entender que el proceso de entrenamiento fue satisfactorio y que ha sido capaz de aprender la forma de clasificar las clases de acordes de forma adecuada.

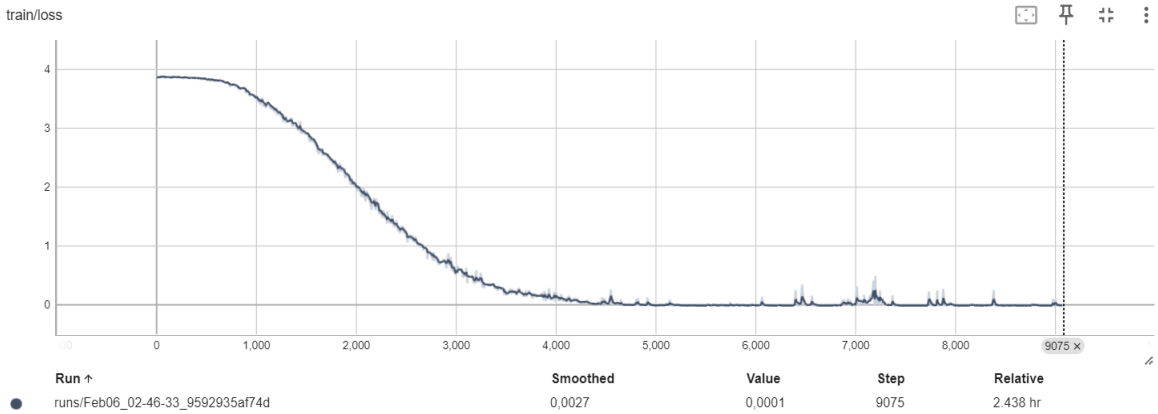

Figura 19. Relación train/loss para el modelo basado en la arquitectura HuBERT. Imagen propia

Dados los resultados de adaptación del modelo, se procedió con las validaciones haciendo uso del 30% de los datos reservados que no fueron empleados para entrenar el modelo tal como se explicó en la sección 4.3. En la Tabla 6 se presentan las métricas asociadas a esta prueba.

Tabla 6. Métricas de validación del modelo desarrollado con la arquitectura HuBERT

Modelo	***Precision***	***Accuracy***	***F1-score***
HuBERT	1	1	1

Una discusión recurrente en ciencia de datos es que los resultados aparentemente perfectos suelen generar incertidumbre, ya que pueden sugerir que el modelo está sobreajustado o que los datos de validación fueron inadvertidamente incluidos en el entrenamiento. Por esta razón, es crucial enfatizar que los datos utilizados en este caso presentan características muy similares entre clases, así como una alta representatividad en cada una de ellas. Esto hace que no sea inusual obtener resultados tan positivos. Sin embargo, estos resultados sugieren de manera preliminar que el modelo está adaptado específicamente a estos datos, pero podría enfrentar dificultades al procesar grabaciones con características significativamente diferentes, como la inclusión de ruido de fondo. Para verificar esta hipótesis, en la Tabla 7 se presentan los resultados de clasificación obtenidos a partir de la segunda base de datos.

Tabla 7. Métricas del modelo desarrollado con la arquitectura HuBERT empleando datos de la base FLstudio

Cantidad de instrumentos	***Precision***	***Accuracy***	***F1-score***
piano/guitarra	0,48	0,48	0,48
piano/guitarra y percusión	0,28	0,28	0,27
piano/guitarra, bajo y percusión	0,01	0,02	0,01

- **Piano/guitarra**: los resultados de esta categoría muestran una precisión del 48%, una exactitud del 48% y un *F1-score* de 0.48, lo que indica un desempeño muy poco aceptable. El modelo no logra adaptarse a las grabaciones que incluyen solo piano y guitarra en un gran porcentaje, probablemente debido a que la mitad de los

datos son interpretados en guitarra y se alejan de lo aprendido con los datos de entrenamiento.

- **Piano/guitarra y percusión**: al agregar percusión, se observa una caída significativa en todas las métricas. La precisión, exactitud y el *F1-score* bajan a valores de 0.28 y 0.27, respectivamente. Esto refleja que la inclusión de un tercer instrumento introduce complejidad que el modelo no es capaz de manejar de manera efectiva, lo que sugiere que no ha sido entrenado con suficiente variedad de combinaciones instrumentales.

- **Piano/guitarra, bajo y percusión**: cuando se incorporan bajo y percusión, los resultados son extremadamente bajos, con una precisión del 1%, una exactitud del 2% y un *F1-score* de 0.01. Estos valores muestran que el modelo prácticamente falla en esta categoría, lo que confirma que su capacidad de generalización es muy limitada frente a escenarios más complejos que difieren radicalmente de los datos de entrenamiento.

Estos resultados confirman las sospechas previamente mencionadas sobre la capacidad del modelo para adaptarse a datos que difieren significativamente de aquellos empleados durante el entrenamiento, donde las grabaciones eran más homogéneas y no contenían ruido de fondo o instrumentos adicionales.

6.2 Resultados Máquina de Vectores de Soporte

El entrenamiento de la *SVM* tuvo dos fases, la primera procesó las caracteristicas extraídas a partir de la arquitectura *HuBERT* y con estas se empleó el método *GridSearch* para encontrar los hiperparámetros que optimizaran la clasificación, en la Tabla 8 se presenta el top 3 de resultados de configuración de parámetros para este escenario.

Tabla 8. Métricas de entrenamiento para la SVM datos caracterizados con HuBERT

***Kernel* y parámetro C**	***Precision***	***Recall***	***F1-score***
C=1 *Kernel = radial*	0,82	0,82	0,81
C=1 *Kernel = linear*	0,82	0,82	0,82
C=10 *Kernel = linear*	0,78	0,78	0,79

Las configuraciones con C=1 y *Kernel* radial o lineal parecen ser las más adecuadas para estos datos, logrando un buen rendimiento. Sin embargo, al aumentar el valor de C o utilizar un *Kernel* más complejo, como el polinómico, el rendimiento del modelo disminuye ligeramente, lo que sugiere que estos ajustes no aportan una mejora significativa y podrían reducir la capacidad de generalización. En términos generales, el *Kernel* lineal es el más consistente, mostrando un buen desempeño en un rango de valores de C.

Empleando la configuración de hiperparámetros C = 1 y *Kernel = linear* se procedió a realizar la validación del modelo a partir del 30% de los datos reservados obteniendo los resultados vistos en la Tabla 9.

Tabla 9. Métricas de validación de la SVM datos caracterizados con HuBERT

Modelo	***Precision***	***Accuracy***	***F1-score***
SVM	0,93	0,93	0,93

Vemos que los resultados de la clasificación de clases por parte de la *SVM* presenta buenos valores tal y como lo vimos con *HuBERT*, durante la evaluación de los resultados vamos a mantener la hipótesis de que el modelo es incapaz de generalizar sus resultados para datos con caracteristicas de entrada significativamente diferentes, esto lo podemos corroborar a partir del set de datos construido en *FLstudio*, en la Tabla 10 se presentan los resultados al realizar la clasificación de acordes con estos datos.

Tabla 10. Métricas de la SVM empleando datos de la base FLstudio caracterizados con HuBERT

Cantidad de instrumentos	***Precision***	***Accuracy***	***F1-score***
piano/guitarra	0,42	0,41	0,41
piano/guitarra y percusión	0,19	0,19	0,19
piano/guitarra, bajo y percusión	0,01	0,02	0,01

Una vez más se corrobora que el modelo a partir de las características extraídas con la arquitectura *HuBERT* no presenta buenos resultados a la hora de procesar grabaciones significativamente diferentes a las entregadas para entrenamiento.

Teniendo en cuenta esta situación, y como segunda fase, se empleó un segundo extractor de características con el fin de validar si el problema se encuentra en los modelos o en la caracterización, en la Tabla 11 se presenta el top 3 de los resultados de entrenamiento empleando las caracteristicas generadas por el modelo *VGGish*.

Tabla 11. Métricas de entrenamiento para la SVM datos caracterizados con VGGish

***Kernel* y parámetro C**	***Precision***	***Recall***	***F1-score***
C=1 *Kernel* = radial	1	1	1
C=1 *Kernel* = linear	1	1	1
C=10 *Kernel* = linear	1	1	1

Con este tipo de caracterización se puede ver que el modelo tiene un rendimiento perfecto, teniendo en cuenta la tendencia de los resultados, esto nos indica que los datos son altamente separables y que en cierta forma las grabaciones que se entregan como entrenamiento no constituyen un reto de abstracción lo suficientemente complejo para los modelos analíticos. En la Tabla 12 se presentan los resultados de validación para el 30% de los datos caracterizados con *VGGish*, en este caso los hiperparámetros usados en el modelo fueron: C = 1 y *Kernel* = linear.

Tabla 12. Métricas de validación SVM para la SVM datos caracterizados con VGGish

Modelo	*Precision*	*Accuracy*	*Accuracy*
SVM	1	1	1

Ahora bien, en la Tabla 13 se presenta el resultado del modelo al momento de clasificar los acordes del segundo set de datos.

Tabla 13. Métricas de la SVM empleando datos de la base FLstudio caracterizados con VGGish

Cantidad de instrumentos	*Precision*	*Accuracy*	*Accuracy*
piano/guitarra	0,63	0,62	0,63
piano/guitarra y percusión	0,46	0,46	0,47
piano/guitarra, bajo y percusión	0,27	0,26	0,26

Vemos un cambio significativo en los resultados a partir de la implementación del caracterizador *VGGish*, esto se debe a que la arquitectura de este modelo fue desarrollada específicamente para extraer de forma robusta, caracteristicas esenciales de los elementos armónicos de una grabación, aun cuando se nota que al modelo le cuesta adaptarse a los datos con alta cantidad de ruido (bajo y percusión juntas), en este caso logra generar algunas clasificaciones correctas, algo que no lograban los modelos anteriores empleando el caracterizador *HuBERT*.

6.3 Resultados *Random Forest*

El entrenamiento del random forest presenta el mismo método que la *SVM*, como primera fase se entrenó el modelo con los datos caracterizados con *HuBERT* y en la segunda con *VGGish*, en la Tabla 14 se presenta el top 3 de los resultados de entrenamiento de la primera fase.

Tabla 14. Métricas de entrenamiento para el random forest datos caracterizados con HuBERT

Parámetros	*Precision*	*Recall*	*Accuracy*
n_estimators = 50, *max_depth* = 10, *max_features* = *'sqrt'*	0,77	0,78	0,77
n_estimators = 50, *max_depth* = 20, *max_features* = *'sqrt'*	0,74	0,74	0,74
n_estimators = 100, *max_depth* = 10, *max_features* = *'auto'*	0,68	0,68	0,68

En esta oportunidad los resultados muestran que el modelo tiene un poco más de dificultades a la hora de realizar la clasificación de los acordes, aun así, las métricas indican una buena capacidad de realizar la tarea por lo que se procede a realizar la validación empleando la configuración *n_estimators* = 50, *max_depth* = 10 y *max_features* = *'sqrt'* con el 30% de datos de validación, los resultados se presentan en la Tabla 15.

Tabla 15. Métricas de validación del modelo Random Forest datos caracterizados con HuBERT

Modelo	*Precision*	*Accuracy*	*Accuracy*
Random Forest	0,91	0,91	0,91

De acuerdo con estos resultados, se puede apreciar como la tendencia de mejora de las métricas del modelo al momento de realizar la validación continúa, al final de los resultados daremos una explicación del por qué las métricas aumentan su valor pasando del entrenamiento a la validación. En la Tabla 16 se presentan los resultados de clasificación del modelo con el set de datos *FLstudio* con el fin de corroborar la hipótesis que se viene manejando desde el inicio de los resultados.

Tabla 16. Métricas del Random Forest empleando datos de la base FLstudio caracterizados con HuBERT

Cantidad de instrumentos	*Precision*	*Accuracy*	*Accuracy*
piano/guitarra	0,34	0,34	0,33
piano/guitarra y percusión	0,08	0,09	0,09
piano/guitarra, bajo y percusión	0,01	0,01	0,01

Al igual que con la *SVM* se corrobora que el modelo empleando las caracteristicas extraídas con la arquitectura *HuBERT* presenta muchas dificultades para realizar una correcta clasificación de los acordes, en este caso se ve una dificultad mayor que la que presenta la *SVM* para estos casos.

En la Tabla 17 se presenta el top 3 de los resultados del entrenamiento del modelo empleando el caracterizador *VGGish*.

Tabla 17. Métricas de entrenamiento para el random forest datos caracterizados con VGGish

Parámetros	*Precision*	Recall	*Accuracy*
n_estimators = 50, *max_depth* = 10, *max_features* = *'auto'*	0,97	0,97	0,97
n_estimators = 50, *max_depth* = 20, *max_features* = *'sqrt'*	0,96	0,95	0,96
n_estimators = 50, *max_depth* = 20, *max_features* = *'auto'*	0,96	0,96	0,96

Teniendo en cuenta los resultados se empleó la configuración *n_estimators* = 50, *max_depth* = 10, *max_features* = *'auto'*, para realizar la validación, ver Tabla 18.

Tabla 18. Métricas de validación Random Forest datos caracterizados con VGGish

Modelo	*Precision*	*Accuracy*	*Accuracy*
Random Forest	0,98	0,98	0,98

En la Tabla 19 se presenta el resultado del modelo al momento de clasificar los acordes del segundo set de datos.

Tabla 19. Métricas del Random Forest empleando datos de la base FLstudio caracterizados con VGGish

Cantidad de instrumentos	*Precision*	*Accuracy*	*Accuracy*
piano/guitarra	0,55	0,54	0,55
piano/guitarra y percusión	0,31	0,31	0,31
piano/guitarra, bajo y percusión	0,15	0,16	0,16

Al igual que en el apartado anterior donde se habla de los resultados de la *SVM*, se puede observar como con el segundo caracterizador el modelo presenta una mejoría significativa, sin embargo, se ve en este caso que le cuesta más lograr buenos resultados.

6.4 Resultados *Gradient Boosting Machine*

El último modelo entrenado fue el *Gradient Boosting Machine* (*GBM*), las condiciones son iguales que la *SVM* o el *Random Forest*, en la Tabla 20 se presenta el top 3 de los resultados de entrenamiento con el caracterizador basado en la arquitectura *HuBERT*.

Tabla 20. Métricas de entrenamiento para GBM datos caracterizados con HuBERT

Parámetros	*Precision*	*Recall*	*Accuracy*
n_estimators = 50, *max_depth* = 4, *learning_rate* = 0.01	0,78	0,78	0,77
n_estimators = 50, *max_depth* = 5, *learning_rate* = 0.01	0,76	0,76	0,76
n_estimators = 100, *max_depth* = 4, *learning_rate* = 0.1	0,71	0,71	0,72

Los resultados muestran que el modelo tiene un comportamiento similar al *random forest*, tiene un poco más de dificultades a la hora de realizar la clasificación de los acordes, pero las métricas presentan una buena capacidad de realizar la categorización de los acordes. La validación, Tabla 21, se realizó empleando la configuración *n_estimators* = 50, *max_depth* = 4, *learning_rate* = 0.0.1

Tabla 21. Métricas de validación del modelo GBM datos caracterizados con HuBERT

Modelo	*Precision*	*Accuracy*	*Accuracy*
GBM	0,94	0,94	0,94

La tendencia finaliza con la constante de mejorar las métricas en el momento de validación. En la Tabla 22 se presentan los resultados de clasificación del modelo con el set de datos *FLstudio*.

Tabla 22. Métricas del modelo GBM empleando datos de la base FLstudio caracterizados con HuBERT

Cantidad de instrumentos	***Precision***	***Accuracy***	***Accuracy***
piano/guitarra	0,63	0,63	0,63
piano/guitarra y percusión	0,2	0,19	0,2
piano/guitarra, bajo y percusión	0,01	0,02	0,01

En esta última prueba se logra corroborar finalmente que si bien los modelos logran diferenciar correctamente los audios con el extractor de caracteristicas basados en *HuBERT* cuando se mantienen con las mismas condiciones, un cambio significativo en las mismas genera fuertes confusiones en el modelo, por lo que podemos inferir que la generalización de una clase requiere de una gran cantidad de datos que contengan dichas variaciones de lo contrario no va a ser un método confiable.

En la Tabla 23 se presenta el top 3 del entrenamiento del modelo empleando el caracterizador *VGGish*.

Tabla 23. Métricas de entrenamiento para GBM datos caracterizados con VGGish

Parámetros	***Precision***	***Recall***	***Accuracy***
n_estimators = 50, *max_depth* = 4, *learning_rate* = 0.1	0,99	0,99	1
n_estimators = 50, *max_depth* = 3, *learning_rate* = 0.1	0,97	0,97	0,97
n_estimators = 50, *max_depth* = 4, *learning_rate* = 0.01	0,97	0,97	0,96

Teniendo en cuenta los resultados se empleó la configuración *n_estimators* = 50, *max_depth* = 4, *learning_rate* = 0.1, para realizar la validación, Tabla 24.

Tabla *24. Métricas de validación GBM datos caracterizados con VGGish*

Modelo	***Precision***	***Accuracy***	***Accuracy***
GBM	1	1	1

En la Tabla 25 se presenta el resultado del modelo al momento de clasificar los acordes del segundo set de datos.

Tabla 25. Métricas del GBM empleando datos de la base FLstudio caracterizados con VGGish

Cantidad de instrumentos	***Precision***	***Accuracy***	***Accuracy***
piano/guitarra	0,7	0,71	0,7
piano/guitarra y percusión	0,51	0,51	0,51
piano/guitarra, bajo y percusión	0,31	0,31	0,32

En estos últimos resultados podemos observar un modelo que de alguna forma muestra capacidades para adaptarse a datos que no presentan el 100% de caracteristicas similares a las de entrenamiento, es claro que el caracterizador es una pieza fundamental dado que al

hacer el cambio en los tres modelos se observa una mejoría significativa, aun así, es evidente que la homogeneidad de datos entre las clases de entrenamiento dificulta significativamente la generalización de los modelos, lo que señala una necesidad de crear bases de datos más robustas que permitan hacer un complemento que mejore el rendimiento del modelo.

Teniendo en cuenta los resultados obtenidos, se puede observar una tendencia clara en la que el modelo *Gradient Boosting* muestra un desempeño superior en comparación con los otros modelos evaluados, incluyendo *HuBERT*, *SVM* y *Random Forest*. El *GBM* logró las mejores métricas de clasificación, especialmente en el grupo de acordes que no formaban parte de la fuente de entrenamiento. Este resultado destaca debido a que las métricas del segundo mejor modelo estuvieron hasta 20 puntos porcentuales por debajo. Estos resultados solo fueron alcanzados al implementar el extractor de características *VGGish*, lo que confirma que esta configuración es la más idónea para el sistema. Además, las pruebas exhaustivas realizadas con los cuatro modelos analíticos refuerzan la solidez de los resultados.

6.5 Discusión

Una vez entrenados los modelos y revisando las métricas, se tienen las siguientes apreciaciones referente al desarrollo de un detector automático de acordes:

- En primer lugar, es importante destacar la paridad en los valores de las métricas aplicadas, es decir, los valores de las métricas se mantuvieron casi sin dispersión al momento de ser calculadas. Esto se debe a que el conjunto de datos está completamente balanceado, lo que explica la similitud entre sus valores.

- En segundo lugar, es notable el aumento en el valor de la métrica de validación con respecto a la obtenida en el momento de entrenamiento. Si el conjunto de datos de prueba fuera significativamente mayor que el conjunto de datos de validación, esto podría ser motivo de preocupación. Sin embargo, en este caso, el conjunto de datos de validación es más extenso, y observamos que los tres modelos clásicos mantienen una tendencia al aumento en el porcentaje de efectividad, lo que sugiere que la cantidad de datos compensa la pérdida prevista en la prueba de entrenamiento.

- En tercer lugar, se destaca cómo la arquitectura *HuBERT* se adapta eficazmente a las características de los datos, siempre y cuando se mantengan con las caracteristicas de entrenamiento. Esta arquitectura logra una eficacia del 100% en las estimaciones tanto en prueba como en validación. Para asegurarnos de la validez de estos resultados, realizamos una validación adicional de los datos para confirmar que el grupo de prueba no se encontrara dentro del grupo de entrenamiento. Además, creamos una función de validación externa a las librerías de *HuggingFace* para verificar predicción por predicción, lo que nos brinda la certeza de que el modelo es capaz de estimar correctamente cualquier acorde

presente en un audio con las características de entrada utilizadas en el entrenamiento.

- En cuarto lugar, el extractor de características es una pieza fundamental a la hora de lograr buenas clasificaciones, en este proyecto se entiende que los datos empleados carecen de diversidad para lograr que los modelos robustos como *HuBERT* logren tener la información suficiente para generalizar los resultados a grabaciones de acordes con elementos de ruido que dificultan su identificación, por eso al emplear un método más robusto como lo es *VGGish* se logró aumentar las métricas de validación e identificar acordes en grabaciones con caracteristicas diferentes a las de entrenamiento.

- Por último, hemos desarrollado una demostración en Gradio que permite estimar el acorde que se reproduce a partir de un archivo de audio de entrada que el usuario puede cargar a través de la interfaz. En la Figura 22, se puede apreciar cómo se realiza en tiempo real la estimación del acorde de Do Mayor a partir del audio que ha sido procesado por el sistema.

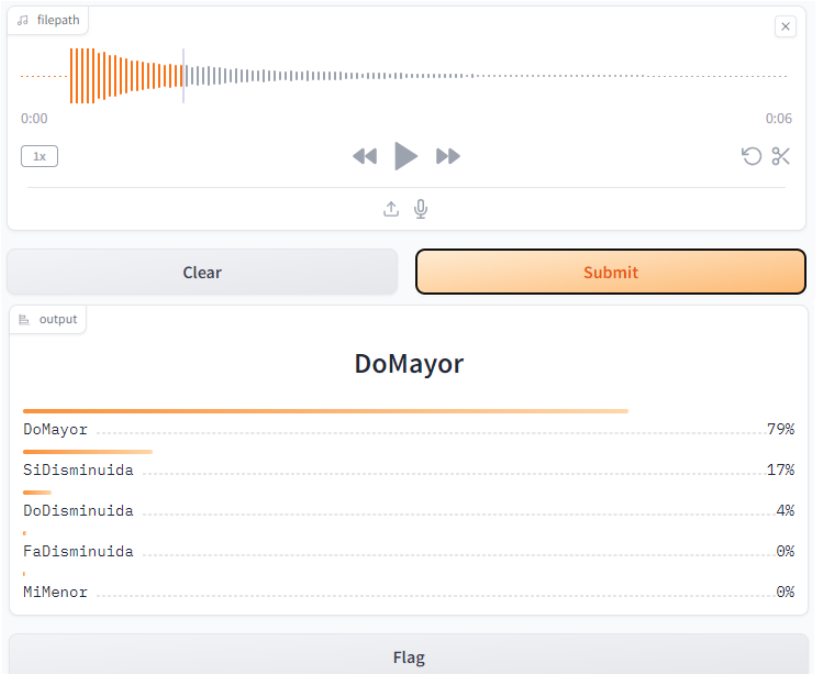

Figura 20. Demo clasificador de acordes con gradio. Imagen propia

7. CONCLUSIONES

En este documento presentamos el desarrollo de un detector automático de acordes utilizando técnicas de caracterización de audios, junto con arquitecturas de modelos analíticos modernos y clásicos que sirvieron como punto de comparación para evaluar el rendimiento de las técnicas empleadas. Según los resultados, tal como se planteó al inicio del documento, presentamos las siguientes conclusiones:

- El principal desafío para lograr una estimación precisa de un acorde musical radica en la correcta extracción de características. En este trabajo, el uso de una arquitectura basada en *Transformers*, como *HuBERT*, permitió que los modelos utilizados alcanzaran métricas superiores al 90%. Esto demuestra la fiabilidad del sistema siempre que los nuevos datos de predicción mantengan las mismas condiciones que los datos de entrenamiento. No obstante, en contextos donde los datos presentan una mayor variabilidad, es necesario emplear modelos más robustos en la caracterización de los audios, como *VGGish*, que ofrece cierta capacidad de compensar, aunque de forma limitada, la falta de una base de entrenamiento más amplia y diversa.

- Los modelos de clasificación actuales son altamente sensibles a los datos con los que han sido entrenados. Cualquier variación en aspectos como la amplitud, el volumen o la frecuencia de muestreo puede impactar de forma significativa en las características que el sistema percibe, lo que a menudo conduce a estimaciones incorrectas. Para mitigar este problema, se requiere una mayor cantidad de datos de entrenamiento que representen estas variaciones, pero hasta ahora, no se dispone de bases de datos suficientemente robustas para abordar eficazmente este desafío.

- Los modelos actuales como *HuBERT* dependen en gran medida tanto de la cantidad como de la calidad de los datos utilizados en el entrenamiento. En este trabajo, se ha observado que, bajo condiciones ideales donde las características de los datos de prueba son muy similares a las del conjunto de entrenamiento, este tipo de modelo genera estimaciones perfectas. Sin embargo, muestra una limitada capacidad de adaptabilidad. Es evidente que el método de caracterización basado en *HuBERT* se ajusta fuertemente a los datos de entrada, considerando el entorno de características como un todo, lo que restringe su capacidad de generalización. En contraste, el algoritmo *VGGish* parece ser más robusto al capturar los fundamentos de la composición, lo que le permite generar mejores estimaciones cuando se enfrenta a datos que presentan características diferentes.

- La validación cruzada, utilizada en este trabajo, fue fundamental para identificar la sensibilidad del modelo ante diferentes configuraciones de datos. Sin embargo, una exploración futura podría incluir la implementación de técnicas de data augmentation que permitan simular condiciones más realistas y ampliar la variabilidad de las muestras, lo cual podría mejorar aún más la robustez y adaptabilidad de los modelos empleados.

7.1 Trabajos futuros.

Para el futuro hay muchos aspectos en los cuales se puede mejorar el detector automático de acordes

- Mejorar la base de datos: los acordes empleados durante el entrenamiento, aunque son de alta calidad, no presentan la cantidad de armónicos formados por instrumentos reales, tampoco representan un ejercicio real de composición musical, por lo que se propone extraer segmentos de interpretación de acordes en canciones producidas, de tal forma en que se sepa su etiqueta, esto lograría enriquecer los datos y dar un ajuste significativo a los modelos modernos como lo es *HuBERT*.

- Interfaz Gráfica: en un mundo donde la tecnología es desbordante y donde los usuarios son cada vez más exigentes, se debe pensar en la experiencia de usuario en lo que hace realmente agradable la interacción entre el individuo objetivo y el producto ofrecido, es así como se deben implementar estrategias *UI/UX* para ambientar de una forma agradable la aplicación y poder presentarlo satisfactoriamente al público.

- Implementar un método de separación de fuentes: dado que una de las mayores dificultades de los modelos fue lograr estimaciones eficaces al momento de incluir ruido de fondo, existen algunos algoritmos que son capaces de extraer el componente armónico de un audio lo que simplificaría la tarea para el detector de acordes.

Con el fin de que se puedan reproducir los escenarios abordados durante este documento, en el repositorio: *https://github.com/alejogil16/chord_detecting.git* podrá encontrar los datos propios creados en *FLStudio,* así como los *notebooks* con los que se entrenó cada modelo, finalmente si desea hacer uso del modelo entrenado con Transformers lo puede encontrar en *HuggingFace* con el nombre: *'alejogil35/distilhubert-finetuned-chorddetection2'*

8. BIBLIOGRAFÍA

A. Radford, K. N. (2018). *Improving language understanding by generative pre-training*.

A. Vaswani, N. S. (2017). *"Attention is all you need" . Advances in neural information processing systems, 30*.

Alvarado Moya, J. P. (2011). *Procesamiento Digital de señales.* Costa Rica: Tecnológico de Costa Rica, Escuela de Ingeniería Electrónica .

Bai, Y. L. (2541-2556). *Efficient data loader for fast sampling-based GNN training on large graphs*. *IEEE Transactions on Parallel and Distributed Systems*, 2021.

Benetos, E. D. (2013). *Automatic music transcription: challenges and future directions. Journal of Intelligent Information Systems. 41, 407-434.*

Betancourt , G. (2005). Las Máquinas de Soporte Vectorial (SVMs). *Scientia et Technica Año XI*.

Breiman, L. (2001). Random forests. *Machine learning*, 5–32.

Brown, J. C. (1991). Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 425-434.

Castro-Ospina, A. E.-S.-E.-V. (2024). Graph-Based Audio Classification Using Pre-Trained Models and Graph Neural Networks. *Sensors*, 2106.

Cherkassky, V. &. (2004). Practical selection of SVM parameters and noise estimation for SVM regression. *Neural networks*, 113-126.

Cortes, C. (1995). Support-Vector Networks. *Machine Learning*.

Davy, A. K. (2007). Signal processing methods for music transcription.

Efron, B. &. (1995). Cross-validation and the bootstrap: Estimating the error rate of a prediction rule. *Stanford, CA, USA: Division of Biostatistics, Stanford University.*, 548-560.

J. Devlin, M.-W. C. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv*.

Jiang, Y. L. (2022). Quo vadis artificial intelligence? *Discover Artificial Intelligence*.

K. P. Bennett and J. Blue. (1998). A support vector machine approach to decision trees. *IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence*, 2396–2401.

Kingsford C, S. S. (2008). What are decision trees? *Nature biotechnology*.

Klapuri, A. (2009). A method for visualizing the pitch content of polyphonic music signals. *ISMIR*, 615–620.

Levine, M. (2003). *The Jazz Piano Book.* Sher Music Co.

Macías , J. (2012). *Diseño e implementacion de un detector automático de acordes.* Madrid: Universidad Carlos III.

Mauch, M. (2010). Automatic chord transcription from audio using computational models of musical context. *Ph.D. dissertation, 2010*.

Paez, A. (2014). Algoritmos Evolutivos y transcripción automática de la Música.

Recuero López , M. (1999). *Ingeniería Acústica* . Madrid, España: Paraninfo.

S. H. Nawab, S. A. (2001). Identification of musical chords using constant-q spectra. *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, 3373–3376.

S. N. Alsubari, S. N. (2022). Data analytics for the identification of fake reviews using supervised learning. *Computers, Materials & Continua*, 3189–3204.
S. Touzani, J. G. (1533–1543). Gradient boosting machine for modeling the energy consumption of commercial buildings. *Energy and Buildings*, 2018.
Saitta, L. (1996). Machine Learning. *Proceedings of the Thirteenth International*.
Sánchez Anzola, N. (2015). Máquinas de soporte vectorial y redes neuronales artificiales en la predicción del movimiento USD/COP spot intradiario. *ODEON*(9), 113-172.
Segura Sogorb, M., & Iñesta Quereda, J. (2015). *Estimación automatica de acordes para uso en la transcripción musical a partir de audio.* Alicante, España: Universidad de Alicante.
Smola , A., & Vishwanathan, S. (2008). *Introduction to Machine Learning* . Cambridge: Cambridge University Press.
W.-N. Hsu, B. B.-H. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 3451–3460.
Zimmermann, A. M. (2006). Recurrent neural networks are universal approximators. *Springer*, 632–640.